

CCIE 职业发展系列

TCP/IP 路由技术 (第一卷)

(第二版)

[美] Jeff Doyle, CCIE #1919 著
Jennifer Carroll, CCIE #1402

葛建立 吴剑章 译

人 民 邮 电 出 版 社
北 京

图书在版编目（**CIP**）数据

TCP/IP 路由技术，第1卷：第2版/（美）多伊尔（Doyle, J.），（美）卡罗尔（Carroll, J.）著；葛建立，吴剑章译．—北京：人民邮电出版社，2007.1（2011.10重印）

（CCIE职业发展系列）

ISBN 978-7-115-15429-3

I. T... II. ①多...②卡...③葛...④吴... III. 计算机网络—通信协议—路由选择

IV. TN915.04

中国版本图书馆CIP数据核字（2006）第125587号

版权声明

Jeff Doyle, Jennifer Carroll: Routing TCP/IP, Volume I, Second Edition
(ISBN: 1587052024)

Copyright © 2006 Cisco Systems, Inc.

Authorized translation from the English language edition published by Cisco Press.

All rights reserved.

本书中文简体字版由美国**Cisco Press**授权人民邮电出版社出版。未经出版者书面许可，对本书任何部分不得以任何方式复制或抄袭。

版权所有，侵权必究。

CCIE职业发展系列

TCP/IP路由技术（第一卷）（第二版）

◆ 著 [美] Jeff Doyle, CCIE#1919
Jennifer Carroll, CCIE#1402

译 葛建立 吴剑章

责任编辑 李 际

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号

邮编 100061 电子邮件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

三河市海波印务有限公司印刷

◆ 开本：787×1092 1/16

印张：48

字数：1208千字

印数：30001-32000册

2007年1月第1版

2011年10月河北第16次印刷

著作权合同登记号 图字：01-2006-0305号

ISBN 978-7-115-15429-3/TP

定价：99.00元

读者服务热线：（010）67132705 印装质量热线：（010）67129223

反盗版热线：（010）67171154

内容提要

本书是一本详细而又完整地介绍互连网络内部网关协议（IGP）的专业书籍，堪称有关IGP方面不可多得的经典之作。本书共分三个部分。第一部分主要介绍了网络和路由选择的基本知识，其中包括IPv4协议、IPv6协议和路由技术。第二部分是本书的精华，这一部分详细、深入地讲述了各种常用的内部路由协议，如RIP、RIPv2、RIPng、无类别路由选择、EIGRP、OSPFv2、OSPFv3、IS-IS等协议，每一章除了对该协议的实现机制和参数详尽阐述，使读者对协议的实现原理有一个清晰的理解外，还通过在实际网络环境中的实例，详细地论述了该协议在Cisco路由器上的配置和故障处理方法，帮助读者获取大量解决实际问题的专业技能。第三部分介绍了如路由重新分配、缺省路由/按需路由选择、路由过滤、路由映射等多种重要而有效的路由控制工具，用来创建和管理多个IP路由选择协议的协调和互操作。附录部分讲述了二进制、十六进制转换、访问列表、CCIE提示等内容。

相对于第一版，本书第二版具有以下更新：在第一版详细讲述IPv4协议中IGP的基础上，大量增加了相应协议在IPv6协议中的实现和配置，其中单独一章用来讲述IPv6中应用的OSPFv3协议，这是本书新版的一大亮点；同时本书根据Internet和Cisco IOS系统的最新发展，适当地删减了如网桥、IGRP等过时的内容，并增加了许多新的IOS增强特性的讲解。

本书的读者不仅是那些准备CCIE考试的考生，也是任何需要完整理解IPv4，特别是IPv6下TCP/IP内部路由选择协议实现的网络设计和工程人员。本书中对协议细节的讲解和对网络实例的探讨相信会让读者受益匪浅。

作者简介

Jeff Doyle (CCIE#1919) 是专注于研究IP路由选择协议、MPLS和IPv6方面的专家。他设计了或协助设计了遍及北美、欧洲、中国、韩国以及日本等很多地区的大型IP服务提供商的网络。Jeff经常在为数众多的研究团体和会议上出现，并在NANOG、JANOG、APRICOT以及IPv6论坛会议上发言。Jeff拥有Memphis州立大学的文学学士学位，并在新墨西哥大学学习了电气工程专业。Jeff目前生活在科罗拉多州的丹佛地区。

Jennifer Carroll (CCIE #1402) 是华盛顿州雷蒙德地区的独立网络顾问专家。在过去的15年中，她设计、实施和优化了许多TCP/IP网络，并开发和讲授了多种有关路由选择协议和Cisco公司路由器方面的网络组网和网间互连课程。读者可以通过电子邮件jennifer.carroll@ieee.org与Jennifer联系。

关于技术审稿人

Frank Knox，是Skyline Computer公司的首席技术执行官，已经供职6年多。他具有两个CCIE资格证书（CCIE#3698：SNA/IP和路由选择/交换），同时也是一位CCSI（Cisco公司授权资深讲师）。除了负责CTO方面的职责外，Frank教授过几门与Cisco有关的高级课程，其中包括为期一周的CCIE实验室考试准备专题培训等。他是与路由器相连的大型主机技术的权威，也是有关网络互连集成方面问题和技术的专家（例如，SNA/IP和语音/数据）。他在IBM公司、GTE（Verizon）Directories公司以及Skyline Computer公司积累了超过37年的网络组网经验。这些经验包括服务、技术支持、产品规划、管理，以及网络互连教育的各个方面。另外，Frank还为Dallas大学电信MBA项目开发和讲授了一些课程。他在Pace大学获得了电信方面的硕士学位（4.0 GPA）。

Steven Edward Moore在Cisco公司工作的6年半的时间里，作为一名工程师担任了不同的角色，目前转到了IP Routing Protocol Scalability团队（IP路由选择协议扩展团队）。在这个团队，他主要是围绕对网络与协议的可扩展性进行扩展的各个方面进行工作：考虑协议架构的新特性和优化，设计对当前协议的可扩展性的测试和评估，帮助客户一起实现对客户网络的功能扩展，以及参加像Networkers会议这样的活动，来指导其他人怎样从路由选择的角度来增强他们的网络性能和可扩展性。

Rena Yang是Cisco公司的一位软件工程师。她具有超过6年的实现Cisco IOS软件编码方面的经验。目前她从事IS-IS协议方面的工作。在这之前，她主要从事IPv4、UDP、访问列表、策略路由和路由选择基础架构方面的工作。Rena在MIT获得了理学学士和计算机科学工程硕士学位。

献词

我愿将本书献给我的妻子Sara和我的孩子们：Anna、Carol、James和Katherine。

——Jeff

我愿将本书献给我的丈夫Mike和我的儿子：Mitchell和Jonathan。他们的耐心和支持帮助我得以完成本书。

——Jennifer

致谢

感谢Brett Bartow、Chris Cleveland、Andrew Cupp、San Dee Phillips，以及Cisco Press的所有工作人员，是他们的工作与努力使本书得以完成。

我们需要感谢本书的技术编辑Steven Moore、Rena Yang和Frank Knox，感谢他们认真细致的编辑工作以及对本书提出的非常重要的忠告和建议。

我们还要感谢Frank Knox、Carl Pike、Chris Tonini和其他Skylabs网络的工作人员。Skylabs lab为我们提供了设备安装和访问方便的实验室，使我们可以完成本书中所有配置和案例研究所需要的工作。

目 录

[Forward for Chinese-Language Edition](#)

[中文版第一版序](#)

[Foreword for the Second Chinese Edition](#)

[中文版第二版序](#)

[原书序](#)

[前言](#)

[第一部分 路由选择的基本知识](#)

[第1章 TCP/IP 回顾](#)

[1.1 TCP/IP协议层](#)

[1.2 IP包头](#)

[1.3 IPv4地址](#)

[1.3.1 首个八位组字节规则](#)

[1.3.2 地址掩码 \(Address Mask\)](#)

[1.3.3 子网和子网掩码](#)

[1.3.4 子网规划](#)

[1.3.5 打破八位组界线](#)

[1.3.6 子网掩码的故障诊断](#)

[1.4 地址解析协议 \(ARP\)](#)

[1.4.1 代理ARP](#)

[1.4.2 无故ARP](#)

[1.4.3 反向ARP](#)

[1.5 ICMP](#)

[1.6 主机到主机层](#)

[1.6.1 TCP](#)

[1.6.2 UDP](#)

[1.7 展 望](#)

[1.8 总结表：第1章命令总结](#)

[1.9 推荐读物](#)

[1.10 复习题](#)

[1.11 配置练习](#)

[1.12 故障诊断练习](#)

[第2章 IPv6概述](#)

[2.1 IPv6地址](#)

[2.1.1 地址表示法](#)

[2.1.2 IPv6的地址类型](#)

[2.2 IPv6包头格式](#)

[2.3 IPv6扩展报头](#)

[2.4 ICMPv6](#)

2.5 邻居发现协议（NDP）

2.5.1 NDP消息

2.5.2 路由器发现（Router Discovery）

2.5.3 地址自动配置（Address Autoconfiguration）

2.5.4 地址冲突检测（Duplicate Address Detection）

2.5.5 邻居地址解析（Neighbor Address Resolution）

2.5.6 私有地址（Privacy Address）

2.5.7 邻居不可到达性的检测

2.6 展望

2.7 复习题

第3章 静态路由

3.1 路由表

3.2 配置静态路由

3.2.1 案例研究：简单IPv4静态路由

3.2.2 案例研究：简单IPv6静态路由

3.2.3 案例研究：汇总路由

3.2.4 案例研究：选择路由

3.2.5 案例研究：浮动静态路由（Floating Static Route）

3.2.6 案例研究：IPv6浮动静态路由

3.2.7 案例研究：均分负载

[3.2.8 案例研究：递归表查询](#)

[3.3 静态路由故障诊断](#)

[3.3.1 案例研究：追踪故障路由](#)

[3.3.2 案例研究：协议冲突](#)

[3.3.3 案例研究：被取代的路由器](#)

[3.3.4 案例研究：追踪IPv6故障路由](#)

[3.4 展望](#)

[3.5 总结表：第3章命令总结](#)

[3.6 复习题](#)

[3.7 配置练习](#)

[3.8 故障诊断练习](#)

[第4章 动态路由选择协议](#)

[4.1 路由选择协议基础](#)

[4.1.1 路径决策](#)

[4.1.2 度量](#)

[4.1.3 收敛](#)

[4.1.4 负载均衡](#)

[4.2 距离矢量路由选择协议](#)

[4.2.1 通用属性](#)

[4.2.2 依照传闻进行路由选择](#)

[4.2.3 路由失效计时器](#)

[4.2.4 水平分隔](#)

[4.2.5 计数到无穷大](#)

[4.2.6 触发更新](#)

[4.2.7 抑制计时器](#)

[4.2.8 异步更新](#)

[4.3 链路状态路由选择协议](#)

[4.3.1 邻居](#)

[4.3.2 链路状态泛洪扩散](#)

[4.3.3 链路状态数据库](#)

[4.3.4 SPF算法](#)

[4.3.5 区域](#)

[4.4 内部和外部网关协议](#)

[4.5 静态或动态路由选择](#)

[4.6 展望](#)

[4.7 推荐读物](#)

[4.8 复习题](#)

[第二部分 内部路由选择协议](#)

[第5章 路由选择信息协议（RIP）](#)

[5.1 RIP的基本原理与实现](#)

[5.1.1 RIP的计时器和稳定性](#)

[5.1.2 RIP消息格式（RIP Message Format）](#)

[5.1.3 请求消息类型（Request Message Type）](#)

[5.1.4 有类别路由选择（Classful Routing）](#)

[5.2 配置RIP](#)

[5.2.1 案例研究：一种基本的RIP配置](#)

[5.2.2 案例研究：被动接口（Passive Interface）](#)

[5.2.3 案例研究：配置单播更新（Unicast Update）](#)

[5.2.4 案例研究：不连续的子网](#)

[5.2.5 案例研究：控制RIP的度量](#)

[5.2.6 案例研究：最小化更新信息的影响](#)

[5.3 RIP故障诊断](#)

[5.4 展望](#)

[5.5 总结表：第5章命令总结](#)

[5.6 推荐读物](#)

[5.7 复习题](#)

[5.8 配置练习](#)

[5.9 故障诊断练习](#)

[第6章 RIPv2、RIPng和无类别路由选择](#)

[6.1 RIPv2的基本原理与实现](#)

[6.1.1 RIPv2的消息格式](#)

[6.1.2 与RIPv1的兼容性](#)

[6.1.3 无类别路由查找](#)

[6.1.4 无类别路由选择协议](#)

[6.1.5 可变长子网掩码（VLSM）](#)

[6.1.6 认证](#)

[6.2 RIPng的基本原理与实现](#)

[6.3 RIPv2的配置](#)

[6.3.1 案例研究：RIPv2的基本配置](#)

[6.3.2 案例研究：与RIPv1的兼容性](#)

[6.3.3 案例研究：使用VLSM](#)

[6.3.4 案例研究：不连续的子网和无类别路由选择](#)

[6.3.5 案例研究：认证](#)

[6.4 RIPng的配置](#)

[6.4.1 案例研究：RIPng的基本配置](#)

[6.4.2 案例研究：RIPng进程的定制](#)

[6.4.3 案例研究：RIPng的度量控制](#)

[6.4.4 案例研究：路由汇总](#)

[6.5 RIPv2与RIPng的故障诊断](#)

[6.6 展 望](#)

[6.7 总结表：第6章命令总结](#)

[6.8 推荐读物](#)

[6.9 复习题](#)

[6.10 配置练习](#)

[6.11 故障诊断练习](#)

[第7章 增强型内部网关路由选择协议（EIGRP）](#)

[7.1 EIGRP的前身：IGRP协议回顾](#)

[7.1.1 进程域](#)

[7.1.2 IGRP的计时器和稳定性](#)

[7.1.3 IGRP的度量](#)

[7.2 从IGRP到EIGRP](#)

[7.3 EIGRP的基本原理与实现](#)

[7.3.1 依赖于协议的模块（Protocol-Dependent Modules）](#)

[7.3.2 可靠传输协议](#)

[7.3.3 邻居发现和恢复](#)

[7.3.4 扩散更新算法](#)

[7.3.5 EIGRP的数据包格式](#)

[7.3.6 地址聚合](#)

[7.3.7 EIGRP和IPv6](#)

[7.4 配置EIGRP](#)

[7.4.1 案例研究：EIGRP的基本配置](#)

[7.4.2 案例研究：非等价负载均衡](#)

[7.4.3 案例研究：设置最大的路径数](#)

[7.4.4 案例研究：多个EIGRP进程](#)

[7.4.5 案例研究：关闭自动路由汇总](#)

[7.4.6 案例研究：末梢路由选择](#)

[7.4.7 案例研究：地址汇总](#)

[7.4.8 认证](#)

[7.5 EIGRP故障诊断](#)

[7.5.1 案例研究：邻居丢失](#)

[7.5.2 “卡”在活动状态的邻居](#)

[7.6 展望](#)

[7.7 总结表：第7章命令总结](#)

[7.8 复习题](#)

[7.9 配置练习](#)

[7.10 故障排除练习](#)

[第8章 开放最短路径优先协议（OSPFv2）](#)

[8.1 OSPF的基本原理与实现](#)

[8.1.1 邻居和邻接关系](#)

[8.1.2 区域（Area）](#)

[8.1.3 链路状态数据库](#)

[8.1.4 路由表](#)

[8.1.5 认证](#)

[8.1.6 按需电路上的OSPF](#)

[8.1.7 OSPF的数据包格式](#)

[8.1.8 OSPF的LSA格式](#)

[8.1.9 可选项字段](#)

[8.2 配置OSPF](#)

[8.2.1 案例研究：一个基本的OSPF配置](#)

[8.2.2 案例研究：使用Loopback接口设置路由器的ID](#)

[8.2.3 案例研究：域名服务查询](#)

[8.2.4 案例研究：OSPF和辅助地址](#)

[8.2.5 案例研究：末梢区域](#)

[8.2.6 案例研究：完全末梢区域](#)

[8.2.7 案例研究：NSSA区域](#)

[8.2.8 案例研究：地址汇总](#)

[8.2.9 案例研究：在区域之间进行过滤](#)

[8.2.10 案例研究：认证](#)

[8.2.11 案例研究：虚链路](#)

[8.2.12 案例研究：运行在NBMA网络上的OSPF](#)

[8.2.13 案例研究：运行在按需电路上的OSPF](#)

[8.3 OSPF故障诊断](#)

[8.3.1 案例研究：孤立的区域](#)

[8.3.2 案例研究：路由汇总配置错误](#)

[8.4 展望](#)

[8.5 总结表：第8章命令总结](#)

[8.6 推荐读物](#)

[8.7 复习题](#)

[8.8 配置练习](#)

[8.9 故障排除练习](#)

[第9章 OSPFv3](#)

[9.1 OSPFv3的基本原理与实现](#)

[9.1.1 OSPFv3与OSPFv2的不同之处](#)

[9.1.2 OSPFv3的消息](#)

[9.1.3 OSPFv3的LSA概述](#)

[9.1.4 OSPFv3的LSA格式](#)

[9.2 OSPFv3的配置](#)

[9.2.1 案例研究：OSPFv3的基本配置](#)

[9.2.2 案例研究：末梢区域](#)

[9.2.3 案例研究：一条链路上的多个实例](#)

[9.2.4 案例研究：运行在NBMA网络上的OSPFv3配置](#)

[9.3 OSPFv3的故障诊断](#)

[9.4 展 望](#)

[9.5 总结表：第9章命令总结](#)

[9.7 复 习 题](#)

[9.8 配置练习](#)

[第10章 集成IS-IS协谈](#)

[10.1 集成IS-IS协议的基本原理与实现](#)

[10.1.1 IS-IS区域](#)

[10.1.2 网络实体标题](#)

[10.1.3 IS-IS的功能结构](#)

[10.1.4 IS-IS的PDU格式](#)

[10.1.5 IS-IS的扩展属性](#)

[10.1.6 泛洪扩散的时延](#)

[10.1.7 提高SPF的效率](#)

[10.2 集成IS-IS协议的配置](#)

[10.2.1 案例研究：IPv4集成IS-IS的基本配置](#)

[10.2.2 案例研究：更改路由器的类型](#)

[10.2.3 案例研究：区域的迁移](#)

[10.2.4 案例研究：路由汇总](#)

[10.2.5 案例研究：认证](#)

[10.2.6 案例研究：IPv6集成IS-IS的基本配置](#)

[10.2.7 案例研究：过渡到多拓扑模式](#)

[10.2.8 案例研究：层之间的路由泄漏](#)

[10.2.9 案例研究：多个L1区域运行在同一台路由器上](#)

[10.3 集成IS-IS协议的故障诊断](#)

[10.3.1 IS-IS邻接关系的故障诊断](#)

[10.3.2 IS-IS链路状态数据库的故障诊断](#)

[10.3.3 案例研究：运行于NBMA网络上的集成IS-IS](#)

[10.4 展 望](#)

[10.5 总结表：第10章命令总结](#)

[10.6 复 习 题](#)

[10.7 配置练习](#)

[10.8 故障诊断练习](#)

[第三部分 路由控制和互操作性](#)

[第11章 路由重新分配](#)

[11.1 重新分配的原则](#)

[11.1.1 度量](#)

[11.1.2 管理距离](#)

[11.1.3 从无类别协议向有类别协议重新分配](#)

[11.2 配置重新分配](#)

[11.2.1 案例研究：重新分配IGRP和RIP](#)

[11.2.2 案例研究：重新分配EIGRP和OSPF](#)

[11.2.3 案例研究：重新分配和路由汇总](#)

[11.2.4 案例研究：重新分配OSPFv3和RIPng](#)

[11.2.5 案例研究：重新分配IS-IS和RIP/RIPng](#)

[11.2.6 案例研究：重新分配静态路由](#)

[11.3 展望](#)

[11.4 总结表：第11章命令总结](#)

[11.5 复习题](#)

[11.6 配置练习](#)

[11.7 故障诊断练习](#)

[第12章 缺省路由和按需路由选择](#)

[12.1 缺省路由基本原理](#)

[12.2 按需路由基本原理](#)

[12.3 配置缺省路由和ODR](#)

[12.3.1 案例研究：静态缺省路由](#)

[12.3.2 案例研究：缺省网络命令](#)

[12.3.3 案例研究：缺省信息始发命令](#)

[12.3.4 案例研究：配置按需路由选择](#)

[12.4 展望](#)

[12.5 总结表：第12章命令总结](#)

[12.6 复习题](#)

[第13章 路由过滤](#)

[13.1 配置路由过滤器](#)

[13.1.1 案例研究：过滤特定路由](#)

[13.1.2 案例研究：路由过滤和重新分配](#)

[13.1.3 案例研究：协议迁移](#)

[13.1.4 案例研究：多个重新分配点](#)

[13.1.5 案例研究：使用管理距离设置路由器优先权](#)

[13.2 展望](#)

[13.3 总结表：第13章命令总结](#)

[13.4 配置练习](#)

[13.5 故障诊断练习](#)

[第14章 路由映射](#)

[14.1 路由映射的基本用途](#)

[14.2 配置路由映射](#)

[14.2.1 案例研究：策略路由选择](#)

[14.2.2 案例研究：策略路由选择和服务质量路由选择](#)

[14.2.3 案例研究：路由映射和重新分配](#)

[14.2.4 案例研究：路由标记](#)

[14.2.5 案例研究：从OSPF路由表中滤掉被标记的路由](#)

[14.2.6 案例研究：使用路由映射重新分配IPv6路由](#)

[14.3 展望](#)

[14.4 总结表：第14章命令总结](#)

[14.5 复习题](#)

[14.6 配置练习](#)

[14.7 故障诊断练习](#)

[第四部分 附 录](#)

Forward for Chinese-Language Edition

Since the publication of *Routing TCP/IP* Volumes I and II, I have had many opportunities to visit the People's Republic of China. In no other country have I received as many warm compliments on these books as I have in China. I am therefore delighted that PT Press is now offering a version of Volume I in Mandarin.

China is aggressively expanding its Internet infrastructure, and along with it services such as mobile IP and online gaming. In the next few years, I predict that China will become the world's leader in the commercial implementation of IPv6. Already China exceeds Japan in the number of PCs, and it will soon have the world's largest mobile network. All of this expansion means that over the coming decade, there will be an enormous increase in the demand for IP networking experts. The Cisco Certified Internet Expert program provides the opportunity for potential employers to discern the best networking engineers, and it provides those holding the certification a testament to their expertise. Therefore the CCIE program plays an increasingly important role in the growing Chinese networking industry. I hope that you, the reader, will find this book useful in your preparation to earn this coveted certification.

Best Regards,

Jeff Doyle

A handwritten signature in black ink, appearing to read "Jeff Doyle". The signature is fluid and cursive, with the first name "Jeff" and last name "Doyle" clearly distinguishable.

中文版第一版序

自从《TCP/IP路由技术》第一卷和第二卷出版以来，我有了很多机会可以来到中国。这些书在其他国家都没有像在中国这样受到大家的热情关注。因此，我很高兴人民邮电出版社能够在中国出版本书第一卷的中文版本。

在中国，Internet基础设施正在迅猛发展与普及，而且也提供了诸如移动IP和在线游戏之类的服务。在未来的几年，我预计中国将成为IPv6商业化部署的世界领导者。在PC机的拥有量上，中国已经超过了日本，并且在不远的将来，中国会拥有世界上最大的移动网络。所有这些巨大的发展都意味着在将来的10年里，中国对IP网络专家的需求量将会有很大的增长。Cisco认证互联网络专家（CCIE）计划可以为企业管理者提供一个渠道，以便识别出最优秀的互联网络工程师；同时，它也为专业技能的证明提供了确实的依据。因此，CCIE计划在中国互联网络产业的发展中将扮演日益重要的角色。我非常期待读者在获取令人羡慕的认证准备过程中，能从本书获益。

最诚挚的问候

Jeff Doyle

A handwritten signature in black ink, reading "Jeff Doyle". The signature is stylized with a cursive script, featuring a large, flowing "J" and "D".

Foreword for the Second Chinese Edition

As I mentioned in the foreword of the first Chinese version of this volume, I make several trips each year to the People's Republic of China and have made many friends there; on every trip I meet people who have kind words about *Routing TCP/IP* Volumes I and II. I am, therefore, very pleased to introduce this Mandarin version of the second edition of Volume I.

Although the first edition of Volume I was first published more than eight years ago, the material it covers remains as relevant today as it was then. Understanding the concepts and protocols in this book is essential for building a foundation of knowledge before progressing to more advanced networking concepts. Too often, engineers try to tackle such difficult topics as IP multicast, complex inter-domain peering, routing policies, and MPLS traffic engineering without first acquiring a firm grasp of the routing basics underlying all of them all. This book, if you study it carefully, will give you that necessary foundation and help you to confidently move on to more advanced subjects.

In the foreword to the previous edition I made several predictions that have since become reality. For example, China now has the world's largest mobile network, which continues to grow at a tremendous rate. I also predicted that China would become the world leader in IPv6 development and deployment. Through the China Next-Generation Internet (CNGI) project, that prediction is quickly moving to reality. There is no other country in the world for which IPv6 is such an urgent and essential protocol, and therefore the coverage of IPv6 in this second edition is of particular importance to Chinese readers.

The translation of any book into another language is a daunting project, and I would like to offer my warmest gratitude to Mr. Ge Jian Li for his hard work in making this Mandarin version available to you.

I would also like to thank Mr. Liu Tao and PT Press for making this

translation possible.

And finally I would like to thank you, dear reader, and the thousands of other Chinese readers who have made *Routing TCP/IP* such a success in China.

Best Regards,

Jeff Doyle

Denver, Colorado

A handwritten signature in black ink, reading "Jeff Doyle". The signature is written in a cursive, flowing style with a large initial "J" and a long, sweeping underline.

中文版第二版序

正如我在本卷第一版的中文版序言中所提到的，我每年都会到中国几次，并在中国结识了很多朋友；每次我都遇到一些提及有关《TCP/IP路由技术》卷一和卷二的朋友。现在我非常高兴地把卷一第二版的中文版介绍给大家。

虽然本书第一版的首次出版距今已有8个年头，但它涵盖的一些内容到现在依然实用。了解本书中提及的概念和协议是建立基本知识架构的基础，这将有利于进一步学习更高级的网络知识。我们常常可以看到，一些网络工程师在未牢固掌握路由技术的基本知识之前，就试图去解决诸如IP多播路由、复杂的域间对等体、路由策略以及MPLS流量工程等困难的课题，而这些高级课题都是建立在前者基础之上的。读者如果认真学习本书，将会获取必要的基本知识，从而可以帮助你信心进一步学习更高级的课题。

在本书第一版的序言中，我的几个预言现在已经变成了现实。例如，中国现在已经具有世界上最大的移动通信网络，并且仍在继续高速发展。我也预言了中国将成为发展和部署IPv6网络的全球领导者。通过中国下一代因特网的项目（CNGI）可以看出，这个预言也将很快变成现实。现今世界上没有哪个国家比中国更迫切需要IPv6协议，因此，在本书第二版中讲述的IPv6技术，对于中国的读者来说显得尤其重要。

把任何一本书翻译成另一种语言都是一件令人生畏的工作，我对本书译者葛建立先生致以最诚挚的谢意，是他所做的努力工作使本书的中文版本得以与读者见面。

我也要感谢人民邮电出版社和刘涛先生使本书中文版的翻译项目变成现实。

最后，我想感谢您，亲爱的读者朋友，以及使《TCP/IP路由技术》在中国如此成功的其他中国读者。

最诚挚的问候

Jeff Doyle

美国科罗拉多州，丹佛

A handwritten signature in black ink, reading "Jeff Doyle". The signature is written in a cursive style with a large, stylized "J" and "D". The background is a light gray rectangular area.

原书序

1976年，当我在数字设备公司（DEC）第一次看到Arpanet IMP的时候，今天我们熟知的网络当时还处于发展初期。SNA、XNS和DECnet等网络处于早期发展阶段，当时有关分组交换和电路交换的讨论还是热点话题。我们这些从事交换和路由选择算法设计的人处理的是具有64KB内存的路由器（虽然我们那时不这样称呼它们），56KB的数据链路被认为是非常快的，具有256个节点的网络就已经足够大了。假如你是卖出了那256台计算机的销售人员，你将会拥有可观的财富并可以退休了。

30年是很长的一段时间，如今，组成Internet的单个网络就包含了数以千计或数以万计的节点，整个Internet包含了数以亿计的计算机。在我们这一代的网络发展过程中，最显著的一点是基于TCP/IP协议簇的Internet基础并没有大的变化；虽然在这期间，计算机体系结构经历了四代或更多代，操作系统技术经历了整整3代，而传输速度也提高了5个数量级。

然而，我们仍然把分组交换网络中的路由选择看作一种“魔法”，为什么呢？

首先，设计强壮的、可扩展的分布式算法是困难的。虽然我们非常希望网络尽可能的简单，但是不可避免的要遇到一些特别的实际案例、网络优化、特殊的拓扑结构，以及链路技术等各种情况，因此网络的复杂性也在一点一点的增加。由于整个网络铲车式升级（fork lift upgrade）是几乎不可行的，因此，目前同时存在多种技术版本，我们必须维护一个向后兼容的、无缝连接的网络来提供网络服务。当控制数据包路由选择的策略变得越来越复杂的时候，我们设计用来自动发现和配置网络的能力就会受到限制，我们又不得不退回到依赖于手工配置和调整性能的技术。最后，当这些网络运行的环境已经从相互之间默认为信任关系的环境发展为会受到内外部攻击的环境时，设计和部署更加安全可靠的路由选择系统就变成一种迫切需要解决的优先问题。

本书完全解开了这个“魔法”之谜。新版的第一卷涵盖了TCP/IP网络所有必不可少的基础知识，并给出了理解在Internet的某个单一管理区域内如何完成路由选择所需要的所有工具。首先，在开始的有关地址和静态路

由的章节中介绍了分组交换网络中路由技术的基本概念，然后深入地讨论了目前最流行的IGP——RIP、EIGRP、OSPF以及IS-IS协议。最后讲述了路由重新分配、路由过滤和策略路由选择等方面的高级课题。

这次出版的第二版也增加了有关IPv6方面的基本内容，并提供了Cisco IOS软件系统最新版本有关示例和配置的所有最新内容。

本书对于任何希望全面了解在TCP/IP网络中路由选择是如何实现的读者都是有帮助的：从路由选择算法的设计原则、地址规划的发展到设计和配置大型自主系统网络路由选择的实践等。

David Oran

Cisco Fellow

前言

路由技术即使在最小的数据通信网络中也是基本的要素。在某种程度上，路由技术和路由器的配置是相当简单的。但是，当网络的规模越来越大，并且越来越复杂的时候，路由选择问题就变得比较突出和难以控制了。或许，有点不恰当地说，作为一名网络系统顾问，我应该感谢当前出现的大规模路由技术难题，这些问题给了我谋生的手段。假设没有它们，“你何以为生？”这句习语可能就会不幸地成为我每天生活词汇的一部分了。

Cisco认证互联网专家（CCIE）在大型网络的设计、故障排除和管理能力方面得到广泛的认同。这种广泛的认同来自于这样一个事实：一个网络工作人员仅仅依赖参加一些课程的培训，并反复依赖记忆一些书面测试的内容是不可能成为一名CCIE的。一名CCIE必须通过一个众所周知、难度非常大的、并且需要亲自动手操作的实验室考试，从而使他或者她的专业技能得到提高。

本书的目标

本书是专门讨论TCP/IP路由问题的两卷书中的第一本。在早期撰写本书的第一版时，Cisco Systems的前CCIE项目经理Kim Lew说过：“我们的目标是使人们成为CCIE，而不是使人们通过CCIE实验室考试。”作者完全赞同这种观点，并且把它作为一种指导原则贯穿到本书的写作当中。虽然这本书包括了很多案例研究和练习可以帮助读者准备CCIE实验室考试，但是作者的主要目的还是提高读者对IP路由技术的理解——能有一个普通的水平并能够在Cisco的路由器上进行实现。

读者对象

本书的读者可以是任何需要完整理解TCP/IP内部路由选择协议的网络设计人员、管理人员或者工程人员。虽然本书的具体实践方面针对Cisco IOS，但是本书的内容也可以应用于任何路由选择平台。

这本书不仅仅是写给那些计划成为Cisco认证互联网专家的读者阅读

的，而且是写给任何希望提高自己的TCP/IP路由技能的读者。这些读者可以划分为以下三类：

- “初学者”——具有基本的网络知识，并且希望开始深入学习网络互连的读者；
- 中级水平的网络专业人员——具有一定的路由器（Cisco 或其他厂商的产品）操作经验，并且计划提高自己的技能达到专家水平的读者；
- 经验丰富的网络专家——这些读者具有丰富和广泛的Cisco路由器的实践经验和专业技能，并且准备参加CCIE实验室考试。但是，这类读者需要自己制定一个复习表和一系列检验与确认自己技能的练习。

本书主要面向具有中级水平的网络专业人员。同时，对于初学者，本书提供了一个网络基本知识的概要。而对于网络方面的专家而言，本书也提供了一些磨炼他们的专业技能所需要的挑战性内容。

对第一版所做的改动

第二版所进行的改动受到几个因素的影响。第一个因素是CCIE本身。当笔者撰写本书的第一版的时候，CCIE——现在称为路由选择与交换专业的CCIE——还是Cisco公司提供的惟一的认证考试。目前，已经形成了将CCIE作为塔尖的认证途径的一系列认证。此外，标准的网络互连专业人员也要比1997年具备更多的知识。考虑到这一点，我们删除了原书的第一章，其中包括网桥、路由器和网络地址的最基本的概念（读者在网络中看到网桥设备的最后时间是什么时候？）。

影响本书第二版所做变化的第二个因素是Cisco公司的IOS软件系统的变化。在撰写本书第一版的时候，IGRP协议还经常使用，而现在它已经是一个过时的协议，存在的主要意义就是作为EIGRP协议的前身。因此，在第二版中删除了有关IGRP协议的章节，并在讲述EIGRP协议的章节中将IGRP协议作为历史回顾的一部分来讲述。IOS软件系统的命令集本身也进行了扩展，以便适应新的功能和选项；我们也增加了在20世纪90年代后期还不存在的协议扩展和命令。

最后，IPv6协议在1997年还主要是建议草案，现在却已经处于全球部署

的早期阶段了。读者需要了解有关这个协议更为细节的知识，以及在不远的将来支持IPv6的IP路由选择协议扩展方面所需要的知识；如果读者还没有准备好，那么本书深入地探讨了IPv6的路由选择技术。

这个版本的其他一些变化是语法和语义方面的。例如，在第一版中，笔者对作为数据链路的“网络”和作为由路由器连接的网络集的“互连网络”进行了区分。虽然这些术语的确很精确，但也有些呆板，现在已经很少使用“互连网络”这样的术语了。相反，“网络”经常用来表示从本地链路到像Level 3、NTT和Sprint这样的全球自主系统的网络。我们尝试在这个新的版本中带给读者比较现代和通用的术语描述。

本书的内容组织

本书共有14章，分为3个部分。

第一部分“路由选择的基本知识”介绍了IPv4协议和IPv6协议的基本知识，以及路由技术的基本概念。虽然一些水平较高的读者可能希望跳过第1章，但是我建议这些读者至少应该浏览一下第3章“静态路由”和第4章“动态路由选择协议”的内容。当然，如果读者还不熟悉IPv6协议的话，也必须阅读第2章“IPv6介绍”。

第二部分“内部路由选择协议”包括了IP路由选择的各种内部网关协议。针对具体协议的每一章都是从该协议的基本原理、实现机制以及参数讲解开始的，并在读者对该协议有了一个总体的了解后，接着通过多个不同的网络拓扑环境中的案例研究，详细地讲述了该协议在Cisco路由器上的配置和故障诊断方法。

外部网关协议，BGP协议，还有组播路由选择、服务质量保证、路由器的安全与管理以及网络地址转换等一些主题，将在“TCP/IP路由技术第二卷”中介绍。

第三部分“路由控制和互操作性”介绍了多种有效的工具，用来创建和管理多个IP路由选择协议的互操作性，例如缺省路由、路由过滤等。同样地，最后部分的这些章节对创建复杂的路由选择策略所需的必要工具做了一个初步介绍，这些策略将会在本书卷二中详细介绍。这些章节和第二部分的章节一样，也是先从概念开始讲解，并以案例研究作为结束。

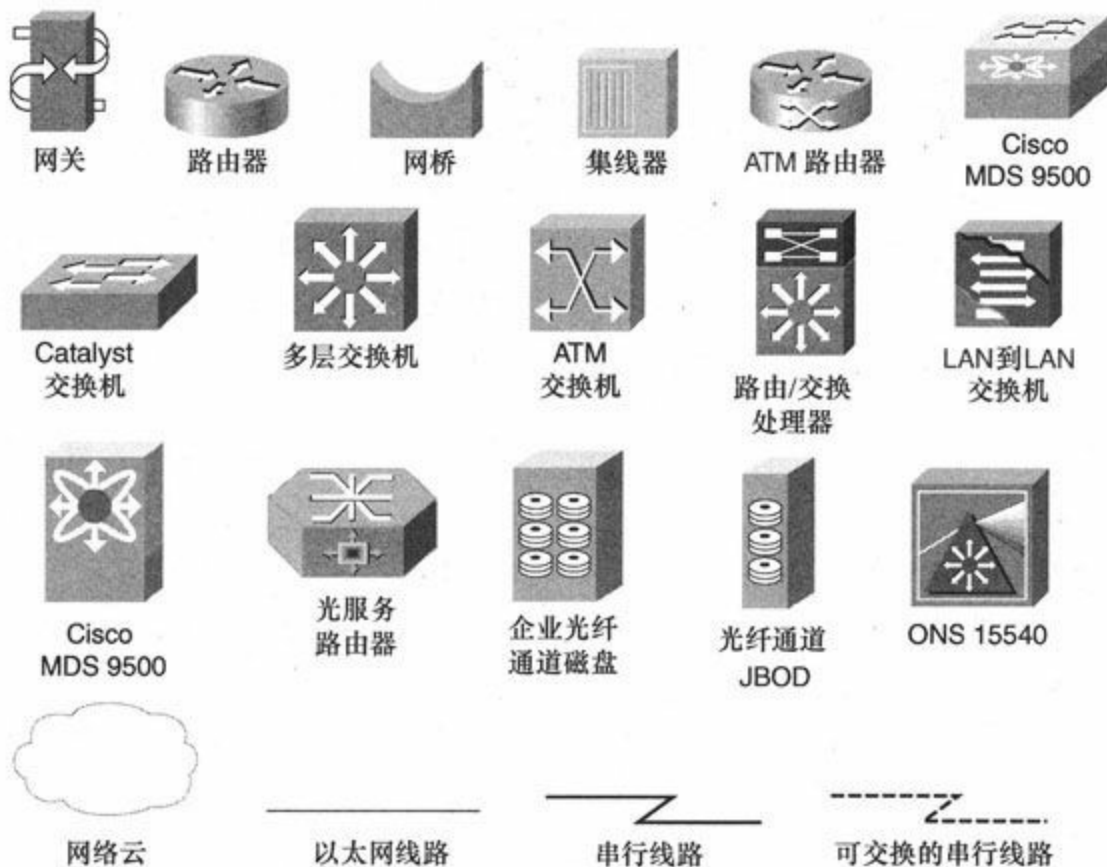
本书的风格

大多数章节在结束时都配有一组复习题、配置练习和故障诊断练习。复习题主要侧重于每章主题的基本理论方面，而配置和故障诊断练习主要侧重于每章主题在Cisco设备上的实际实现。

在每章末尾还列出了一张命令总结表，简要介绍了在这一章中使用到的Cisco IOS中的所有重要命令。这些命令使用的惯例和Cisco IOS命令参考中使用的惯例是一样的。



本书使用的图标



命令语法定义

本书在介绍命令语法时使用的约定与《IOS命令参考手册》相同，这些约定如下：

- 粗体字 表示实际需要键入的命令和关键字，在实际的配置示例和输出信息（不是一般的命令语法）中，粗体字表示用户手工输入的命令（例如**show**命令）；
- 斜体字 表示需要用实际数值替换的参数；
- 竖线（|）表示在几个选项中选择一项，并且这些项是互相排斥的；
- 方括号[]表示可选的参数；
- 大括号{}表示一个必需的选项；

- 方括号内嵌大括号[{}]表示在一个可选项中的必选项。

译者注：本书第一版出版后，受到了广大读者的关注与厚爱，译者收到许多读者的来信，本书新版对第一版中出现的错误进行了修正，在此感谢读者的批评指正，欢迎广大读者来信交流，译者E-mail: cfa35@126.com。

第一部分

路由选择的基本知识

第1章 ICP/IP回顾

第2章 IPv6概述

第3章 静态路由

第4章 动态路由选择协议

本章包括以下主题：

- TCP/IP协议层；
- IP包头（IP Packet Header）；
- IPv4地址；
- 地址解析协议（ARP）；
- Internet控制消息协议（ICMP）；
- 主机到主机层。

第1章

TCP/IP 回顾

考虑到这本书的书名是《TCP/IP路由技术》，有必要从回顾TCP/IP的基本知识开始讲起，然后再讲述如何进行TCP/IP路由选择。如果读者正在准备Cisco认证互连网专家（Cisco Certified Internetwork Expert, CCIE）的考试，或者仅仅把本书作为路由选择技术方面的参考书，那么读者大概已经了解了本章所要讲述的大部分内容。但是，如果读者阅读一下本章，作为对基本知识的复习也不是一件坏事，有时还会有所帮助。

本章主要回顾了启用、控制或帮助TCP/IP路由选择的协议，对TCP/IP协议簇并不作深入讨论。本章最后列出的几本参考读物均对TCP/IP进行了深入详细的讲解，读者可以至少选择其中的一本进行阅读。

早在20世纪70年代初期，Vint Cerf和Bob Kahn就对TCP/IP及其分层协议框架进行了构思，它的提出先于ISO的OSI参考模型。本章对TCP/IP协议层的简单回顾有助于读者理解TCP/IP的多种功能与服务是如何进行相互关联的。

1.1 TCP/IP协议层

图1-1展示了TCP/IP协议簇与OSI参考模型的相互关系。[\[1\]](#)在TCP/IP协议簇中，网络接口层对应于OSI的物理和数据链路层，但实际上在规范中并不存在这一层。如图1-1所示，作为对物理和数据链路层的表示，它已经成为事实上的一个层次。在本节中，我们将使用OSI的术语——物理和 数据链路层来描述它。

OSI	TCP/IP
应用层	应用层
表示层	
会话层	
传输层	主机到主机层
网络层	Internet 层
数据链路层	网络接口层
物理层	

图1-1 TCP/IP协议簇

物理层包含了多种与物理介质相关的协议，这些物理介质用以支撑

TCP/IP通信。物理层的协议按照正式的分类可以分为4类，这4类涵盖了物理介质的所有方面：

- 电子/光学协议 ——描述了信号的各种特性。例如，电压或光强度、位定时、编码和信号波形。
- 机械协议 ——规定了连接器的尺寸或导线的金属成分。
- 功能性协议 ——描述了做什么。例如，在EIA-232-D连接器第4管脚上的功能描述是“请求发送”。
- 程序性协议 ——描述了如何做。例如，在EIA-232-D导线上，二进制1表示电压小于-3V。

数据链路层 包含了控制物理层的协议：如何访问和共享介质、怎样标识介质上的设备，以及在介质上发送数据之前如何完成数据成帧。典型的数据链路协议有IEEE 802.3/以太网、帧中继、ATM以及SONET。

Internet 层 与OSI的网络层相对应，主要负责定义数据包格式和地址格式，为经过逻辑网络路径的数据进行路由选择。当然，网络层也是本书内容涉及最多的一层。

与OSI传输层相对应的是主机到主机层，它指定了控制Internet层的协议，这就像数据链路层控制物理层一样。主机到主机层和数据链路层都定义了流控和差错控制机制。二者不同之处在于，数据链路层协议强调控制数据链路路上的流量，即连接两台设备的物理介质上的流量；而传输层控制逻辑链路路上的流量，即两台设备的端到端连接，这种逻辑连接可能跨越一连串数据链路。

应用层 与OSI的会话层、表示层、应用层相对应。虽然一些路由选择协议使用这一层，例如边界网关协议（BGP）、路由选择信息协议（RIP）等，[\[2\]](#)但是应用层最常用的服务是向用户应用提供访问网络的接口。

对于图1-1中所示的协议簇和其他协议簇来说，各层之间多路复用是一个通用功能。许多应用可以使用主机到主机层的一个服务，同样许多主机到主机层的服务也可以使用Internet层。多个协议簇（如IP、IPX、AppleTalk）还可以通过数据链路协议共享一条物理链路。

1.2 IP包头

图1-2给出了IP包头（Packet Header）的格式，相应标准见RFC791。数据包中的大多

数字段对路由选择都很重要。

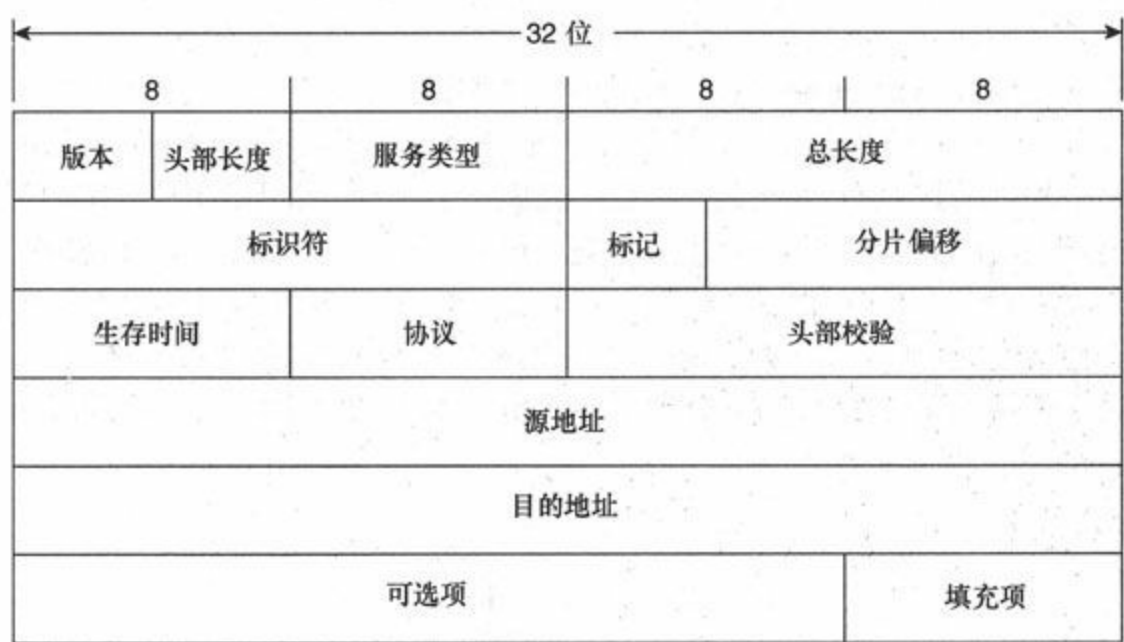


图1-2 IP数据包协议

- 版本（**Version**）——标识了数据包的IP版本号。这个4位字段的值设置为二进制的0100表示IP版本4（IPv4），设置为0110表示IP版本6（IPv6）。本章主要涉及的是IPv4，在第2章中主要讲述IPv6。在表1-1中列出了所有已分配的现行版本号及相关RFC。除4和6（早期提出的简单Internet协议——即SIP协议，也使用版本号6）之外，其他的所有版本号仅作为“历史产物”而存在，感兴趣的读者可以阅读相关的RFC。

表1-1 IP版本号

版本号	版本	RFC
-----	----	-----

0	保留	
1~3	未分配	
4	Internet协议版本 4 (IPv4)	791
5	ST数据报 (Datagram) 模式	1190
6	简单Internet协议 (SIP)	
6	IPv6	1883
7	TP/IX	1475
8	P Internet协议 (PIP)	1621
9	使用更大地址的TCP和UDP (TUBA)	1347
10~14	未分配	
15	保留	

- 报头长度 (**header length**) —— 字段长度为4位，正如字段名所示，它表示32位字长的IP报头长度。设计报头长度字段是因为数据包的可选项字段 (在本节后面部分将会讨论) 的大小会发生变化。IP报头最小长度为20个八位组，最大可以扩展到60个八位组——通过这个字段也可以描述32位字的最大长度。

- 服务类型 (**Type of Service, ToS**) —— 字段长度为8位，它用来指定特殊的数据包处理方式。服务类型字段实际上被划分为两个子字段：优先权和ToS。优先权用来设置数据包的优先级，这就像邮寄包裹一样，可以是平信、隔日送到或两日内送到。ToS允许按照吞吐量、时延、可靠性和费用方式选择传输服务。虽然ToS字段通常不用 (所有位均被设置为 0)，但是在开放式最短路径优先 (OSPF) 协议的早期规范中还是称为ToS路由选择的。优先权位偶尔在服务质量 (QoS) 应用中使用。图1-3的a部分简要地说明了8个ToS位，更详细的信息可以参见RFC1340和RFC1349。

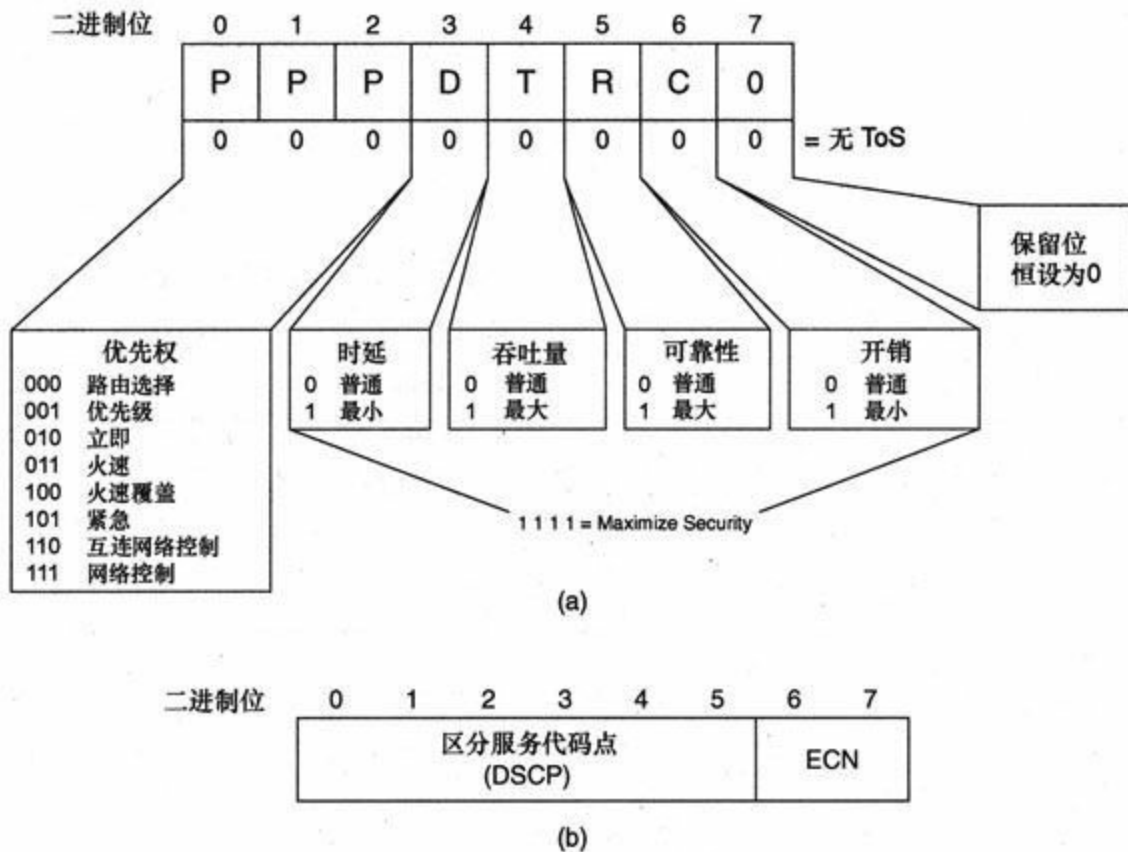


图1-3 服务类型字段 (a) 或区分服务和ECN (b) 字段

在最近几年，ToS字段已经作为区分服务（Diffserv）架构的一部分被重新定义了。[\[3\]](#) 区分服务架构为IP数据包所创建的处理比通过相对严格的ToS定义所允许的处理灵活得多。在DiffServ下，我们能够在在一台路由器上定义服务分类，将数据包归类到这些分类中去。路由器可以根据它们的分类使用不同的优先级对数据包进行排序和转发。每一个排序和转发的处理称为一个*Per-Hop Behavior*（PHB）。虽然DiffServ定义了这个架构或体系，但这个机制本身称为区分服务类别或简单地称为服务类别（CoS）。

图1-3中的（b）部分显示了ToS字段是如何重新定义的，开始的6个位现在构成了区分代码点（DiffServ Code Point, DSCP）。利用这6个位，我们可以使用任意数值或根据在区分服务体系结构中预先定义的服务类别，最多可以定义64个不同的服务类别，并可整理到PHB中。请注意，在IP报头中的这个字段保留了8位；区分服务体系结构重新定义了路由器对这个字段中数值的解释。

- 显式拥塞通知（**Explicit Congestion Notification, ECN**）——在图1-3中的显式拥塞通知是某些路由器用来支持显式拥塞通知的，当它支持该特性时，这些位可以用于拥塞信号（ECN=11）。

[\[4\]](#)

- 总长度（**Total Length**）——数据包总长度字段的长度为16位，以八位组为单位计，其中包括IP报头。接收者用IP数据包总长度减去IP报头长度，就可以确定数据包数据有效载荷的大小。16位长的二进制数用十进制表示最大可以为65535，所以IP数据包的最大长度是65535。

- 标识符（**Identifier**）——字段长度为16位，通常与标记字段和分段偏移字段一起用于数据包的分段。如果数据包原始长度超过数据包所要经过的数据链路的最大传输单元（MTU），那么必须将数据包分段为更小的数据包。例如，一个大小为5000字节的数据包在穿过网络时，如果遇到一条MTU为1500字节的数据链路，即数据帧最多容纳大小为1500字节的数据包。路由器需要在数据成帧之前将数据包分段成多个数据包，其中每个数据包长度不得超过1500字节；然后路由器在每片数据包的标识字段上打上相同的标记，以便接收设备可以识别出属于一个数据包的分段。 [\[5\]](#)

- 标记字段（**Flag**）——长度为3位，其中第1位没有使用。第2位是不分段（DF）位。当DF位被设置为1时，表示路由器不能对数据包进行分段处理。如果数据包由于不能被分段而未能被转发，那么路由器将丢弃该数据包并向源点发送错误消息。这一功能可以在网络上用于测试MTU值。参见示例1-1所示，在IOS软件系统中，使用扩展Ping工具可以对DF位进行设置。

示例1-1 为了测试穿越网络的MTU值，IOS软件中的扩展Ping工具允许设置DF位。在ping的输出信息中，到达目的地172.16.113.17的路径的最大MTU为1478字节

Handy#**ping**

Protocol [ip]:

Target IP address: **172.16.113.17**

Repeat count [5]: 1

Datagram size [100]:

Timeout in seconds [2]:

Extended commands [n]: **y**

Source address:

(待续)


```

Type of service [0]:
Set DF bit in IP header? [no]: y
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: r
Number of hops [9]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
Sweep range of sizes [n]: y
Sweep min size [76]: 500
Sweep max size [18024]: 2000
Sweep interval [1]: 500
Type escape sequence to abort.
Sending 4, [500..2000]-byte ICMP Echos to 172.16.113.17, timeout is 2 seconds:
Packet has IP options: Total option bytes= 39, padded length=40
Record route: <*> 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
               0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0

Reply to request 0 (16 ms) (size 500). Received packet has options
Total option bytes= 40, padded length=40
Record route: 172.16.192.5 172.16.113.18 172.16.113.17 172.16.113.17
               172.16.192.6 172.16.192.5 <*> 0.0.0.0 0.0.0.0 0.0.0.0
End of list

Reply to request 1 (24 ms) (size 1000). Received packet has options
Total option bytes= 40, padded length=40
Record route: 172.16.192.5 172.16.113.18 172.16.113.17 172.16.113.17
               172.16.192.6 172.16.192.5 <*> 0.0.0.0 0.0.0.0 0.0.0.0
End of list

Reply to request 2 (28 ms) (size 1500). Received packet has options
Total option bytes= 40, padded length=40
Record route: 172.16.192.5 172.16.113.18 172.16.113.17 172.16.113.17
               172.16.192.6 172.16.192.5 <*> 0.0.0.0 0.0.0.0 0.0.0.0
End of list

Unreachable from 172.16.192.6, maximum MTU 1478 (size 2000).
Received packet has options
Total option bytes= 39, padded length=40
Record route: <*> 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
               0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0

Success rate is 75 percent (3/4), round-trip min/avg/max = 16/22/28 ms
Handy#

```

第3位表示还有更多分段（MF）位，当路由器对数据包进行分段时，除了最后一个分段的MF位设置为0外，其他所有分段的MF位均设置为1，以便接收者直到收到MF位为0的分段为止。

- 分段偏移（**Fragment Offset**）——字段长度为13位，以8个八位组为单位，用于指明分段起始点相对于报头起始点的偏移量。^[6]由于分段到达时可能错序，所以分段偏移字段可以使接收者按照正确的顺序重组数据包。

请注意，如果一个分段在传输中丢失，那么必须在网络中同一点对整个数据包重新分段并重新发送。因此，容易发生故障的数据链路会造成时延不成比例。另外，如果由于网络拥塞而造成分段丢失，那么重传整组分段会进一步加重网络拥塞。

- 生存时间（**Time To Live, TTL**）——字段长度为8位，在最初创建数据包时TTL即被设置为某个特定值。当数据包逐个沿路由器被传输时，每台路由器都会降低TTL的数值。当TTL值减为0时，路由器将会丢弃该数据包并向源点发送错误信息。这种方法可以防止数据包在网络上无休止地被传输。

按照最初构想，TTL值以s（秒）为单位。如果数据包在路由器上被延迟的时间超过1s，路由器将会相应地调整TTL值。然而，这种方法实施起来十分困难，从来没有被广泛地支持。现代的路由器不管实际时延是多少，统统将TTL值减1，所以TTL实际上是表示跳数。^[7]虽然TTL常见的值为15和32，但是建议的缺省值是64。像IOS软件中的**trace**命令这样的一些追踪工具使用的是TTL字段。如果路由器被告知需要追踪到达主机地址为10.11.12.13的路径，路由器将发送3个数据包，其中TTL值被设置为1；第1台路由器将会把TTL值减少到0，而且在丢弃数据包的同时向源点发送错误信息。源点路由器通过阅读错误信息从而得知发送错误信息的路由器即为路径上的第1台路由器。再一次被路由器发送的3个数据包的TTL值被设置为2。第1台路由器将TTL值减1，第2台路由器将TTL值再减1后为0，此时源点路由器将会接收到第2台路由器发送来的错误信息。第3次发送的数据包TTL值被设置为3，依此类推，直到目的地被发现。最终，沿着网络路径所有的路由器都会被标识出来。示例1-2中显示了IOS软件中路由追踪的输出结果。

示例1-2 追踪工具使用**TTL**字段来标识沿途路由器。星号表示超时数

数据包

```
Elvis#traceroute www.cisco.com

Type escape sequence to abort.
Tracing the route to cio-sys.Cisco.COM (192.31.7.130)

 0 172.18.197.17  4 msec  4 msec  4 msec
 1 ltlrichard-s1-13.hwy51.com (172.18.197.1)  36 msec  44 msec  2536 msec
 2 cperkins-rtr-fr2.hwy51.com (10.168.204.3)  104 msec  64 msec  *
 3 cberry.hwy51.com (10.168.193.1)  92 msec  *
 4 jllewis-inner.hwy51.com (10.168.207.59)  44 msec  *  44 msec
 5 bholly-fw-outer-rt.hwy51.com (10.168.207.94)  44 msec  *  48 msec
 6 sl-stk-14-S10/0:6-512k.sprintlink.net (144.228.214.107)  92 msec  *
 7 sl-stk-2-F1/0/0.sprintlink.net (144.228.40.2)  52 msec  1156 msec  *
 8 sl-mae-w-H1/0-T3.sprintlink.net (144.228.10.46)  100 msec  124 msec  2340 msec
 9 sanjose1-br1.bbnplanet.net (198.32.136.19)  2264 msec  164 msec  *
10 paloalto-br2.bbnplanet.net (4.0.1.10)  64 msec  60 msec  *
11 su-pr2.bbnplanet.net (131.119.0.218)  76 msec  76 msec  76 msec
12 cisco.bbnplanet.net (131.119.26.10)  2560 msec  76 msec  936 msec
13 sty.cisco.com (192.31.7.39)  84 msec  72 msec  *
14 cio-sys.Cisco.COM (192.31.7.130)  60 msec  *  64 msec

Elvis#
```

- 协议（**Protocol**）——字段长度为8位，它给出了主机到主机层或传输层协议的“地址”或协议号，协议字段指定了数据包中信息的类型。当前已分配了100多个不同的协议号，表1-2给出了其中一些较常用的协议号。

表1-2 一些众所周知的协议号

协议号	主机到主机层协议
1	Internet消息控制协议（ICMP）
2	Internet组管理协议（IGMP）
4	被IP协议封装的IP
6	传输控制协议（TCP）
17	用户数据报协议（UDP）
45	域间路由选择协议（IDRP）
46	资源预留协议（RSVP）
47	通用路由选择封装（GRE）
54	NBMA下一跳解析协议（NHRP）
88	Cisco Internet网关路由选择协议（IGRP）
89	开放式最短路径优先（OSPF）

- 报头校验和（**Header Checksum**）——是针对IP报头的纠错字段。校验和不计算被封装的数据，UDP、TCP和ICMP都有各自的校验和。报头校验和字段包含一个16位二进制补码和，这是由数据包发送者计算得到的。接收者将连同原始校验和重新进行16位二进制补码和计算。如果数据包传输中没有发生错误，那么结果应该16位全部为1。回忆前面所述内容，由于每台路由器都会降低数据包的TTL值，所以每台路由器都必须重新计算校验和。RFC1141讨论了一些简化计算的策略。

- 源地址和目的地址（**Source and Destination Address**）——字段长度为32位，分别表示发送者数据包源点和目的地的IP地址。IP地址的格式将会在1.3节中讨论。

- 可选项（**Options**）——是一个长度可变的字段，并像其名字所表示的，它是可选的。可选项被添加在包头中，包括源点产生的信息和其他路由器加入的信息；可选项字段主要用于测试。常用的可选项如下：



松散源路由选择（**Loose Source Routing**）——它给出了一

连串路由器接口的IP地址序列。数据包必须沿着IP地址序列传送，但是允许在相继的两个地址之间跳过多台路由器。

➤ 严格源路由选择（**Strict Source Routing**）——它也给出了一系列路由器接口的IP地址序列。不同于松散源路由选择，数据包必要严格按照路由转发。如果下一跳不再列表中，那么将会发生错误。

➤ 记录路由（**Record Route**）——当数据包离开时为每台路由器提供空间记录数据包的出站接口地址，以便保存数据包经过的所有路由器的记录。记录路由选项提供了类似于路由追踪的功能，但是不同点在于这里记录了双向路径上的出站接口信息。

➤ 时间戳（**Timestamp**）——除了每台路由器还会记录一个时间戳之外，时间戳选项十分类似于记录路由选项，这样数据包不仅可以知道自己到过哪里，而且还可以记录到达的时间。

在Cisco路由器上使用扩展的Ping工具可以调用所有这些选项。示例1-1中使用了记录路由选项，示例1-3使用了松散源路由选择和时间戳选项，严格源路由选择选项在示例1-4中被使用。

示例1-3 使用Cisco的扩展Ping工具来设置IP报头中的可选项字段的各项参数。在这个例子中，用到了松散源路由选择选项和时间戳选项

```
Handy#ping
Protocol [ip]:
Target IP address: 172.16.113.9
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address:
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: 1
Source route: 172.16.113.14 172.16.113.10
Loose, Strict, Record, Timestamp, Verbose[LV]: t
Number of timestamps [ 6 ]: 2
Loose, Strict, Record, Timestamp, Verbose[LTV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.113.9, timeout is 2 seconds:
Packet has IP options: Total option bytes= 23, padded length=24
  Loose source route: <*> 172.16.113.14 172.16.113.10
  Timestamp: Type 0. Overflows: 0 length 12, ptr 5
    >>Current pointer<<
    Time= 0
    Time= 0

Request 0 timed out
Reply to request 1 (76 ms). Received packet has options
  Total option bytes= 24, padded length=24
  Loose source route: 172.16.113.13 172.16.192.6 <*>
  Timestamp: Type 0. Overflows: 6 length 12, ptr 13
    Time= 80FF4798
    Time= 80FF4750
    >>Current pointer<<
  End of list

Request 2 timed out
Reply to request 3 (76 ms). Received packet has options
  Total option bytes= 24, padded length=24
  Loose source route: 172.16.113.13 172.16.192.6 <*>
  Timestamp: Type 0. Overflows: 6 length 12, ptr 13
    Time= 80FF4FC0
    Time= 80FF4F78
    >>Current pointer<<
  End of list

Request 4 timed out
Success rate is 40 percent (2/5), round-trip min/avg/max = 76/76/76 ms
Handy#
```

示例**1-4** 这里使用扩展**Ping**在**ping**数据包中设置严格源路由选择选项

```
Handy#ping
Protocol [ip]:
Target IP address: 172.16.113.10
Repeat count [5]: 2
Datagram size [100]:
```

(待续)

```
Timeout in seconds [2]:
Extended commands [n]: y
Source address:
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: s
Source route: 172.16.192.6 172.16.113.17 172.16.113.10
Loose, Strict, Record, Timestamp, Verbose[SV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 2, 100-byte ICMP Echos to 172.16.113.10, timeout is 2 seconds:
Packet has IP options: Total option bytes= 15, padded length=16
    Strict source route: <*> 172.16.192.6 172.16.113.17 172.16.113.10

Reply to request 0 (80 ms). Received packet has options
    Total option bytes= 16, padded length=16
    Strict source route: 172.16.113.10 172.16.113.17 172.16.192.6 <*>
    End of list

Reply to request 1 (76 ms). Received packet has options
    Total option bytes= 16, padded length=16
    Strict source route: 172.16.113.10 172.16.113.17 172.16.192.6 <*>
    End of list

Success rate is 100 percent (2/2), round-trip min/avg/max = 76/78/80 ms
Handy#
```


- 填充（**Padding**）——该字段通过在可选项字段后面添加0来补足32位，这样保证报头长度是32位的倍数。

示例1-5显示了协议分析器捕获到的IP报头的信息。请与图1-2中的信息作一下比较。

示例**1-5** 在协议分析器的窗口中，可以看到**IP**包头各字段及每个字段的值

```
Internet Protocol, Src Addr: 172.16.1.102 (172.16.1.102), Dst Addr: 224.0.0.5 (224.0.0.5)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00)
  Total Length: 64
  Identification: 0x6e61 (28257)
  Flags: 0x00
  Fragment offset: 0
  Time to live: 1
  Protocol: OSPF IGP (0x59)
  Header checksum: 0xbcc8 (correct)
  Source: 172.16.1.102 (172.16.1.102)
  Destination: 224.0.0.5 (224.0.0.5)
```

1.3 IPv4地址

IPv4地址长度为32位。像所有其他网络层地址一样，IPv4地址也包括网络号和主机号两部分。网络号部分唯一地标识了一条物理链路或逻辑链路，对于与该链路相连的所有设备来说网络号部分是共同的。而主机号部分唯一地标识了该链路上连接的具体设备。

有几种方式可以表示IP地址的32位。举例来说，32位的IP地址00001010110101100101011110000011可以用十进制表示为181819267。

由此可见用二进制表示IP地址十分麻烦，而全部32位数字用十进制格式表示计算起来又很耗时。图1-4给出了一种更好的表示方法。

32位的地址包含4个字节，每个字节均可以用0~255之间的十进制数表示，而每个十进制数之间用点号分隔。在图1-4中，将32位的地址映射到用点分十进制法表示的地址上。[\[8\]](#)

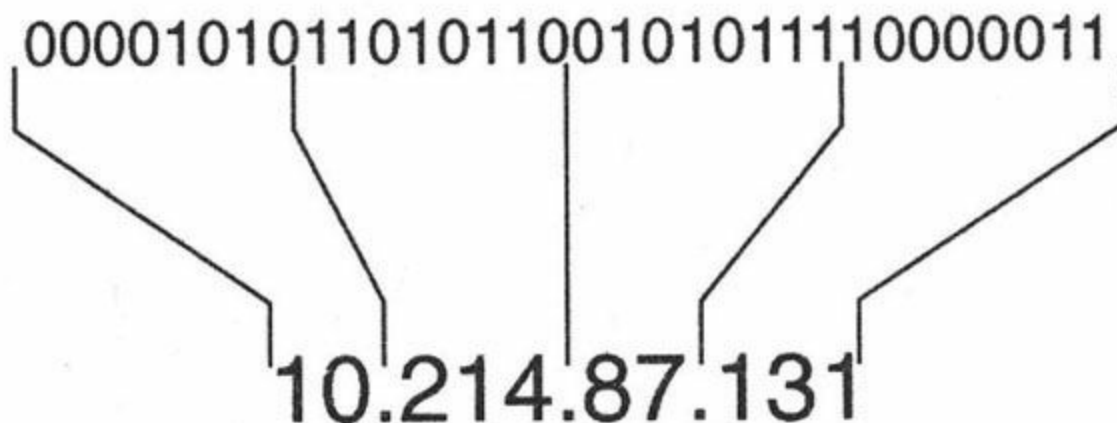


图1-4 虽然使用点分十进制表示法书写IP地址十分方便，但是不要将它与路由器（或主机）所看到的32位字符串混淆起来

在使用IPv4地址时需要记住一点，点分十进制表示法便于人们阅读和书写，而路由器更适合使用32位的二进制串，而不是按照4个八位组的方式读取地址。牢记这一点可以避免许多易犯的错误。如果读者没有接触过二进制数，特别是二进制与十进制之间的转换方法，那么建议读者在继续学习这一章的内容之前可以阅读一下附录A的内容。

IP地址与众不同的特性可能就是，IPv4地址不像其他网络层地址的网络号和主机号是固定不变的，IP地址的网络号和主机号可以在32位的界线内发生变化。也就是说，IP地址的网络号和主机号都有可能占据32位中的大多数位，也可能两者平分32位。例如NetWare和AppleTalk协议，由于它们主要用于相对较小的网络，所以协议的网络层地址的网络号和主机号长度是固定。这样的安排的确使得工作更加容易，接收设备可以从地址中读入固定的位来获取网络号，剩下的位便是主机号。

然而，TCP/IP从最初设计出来到现在可以灵活地应用于任何网络，从很简单的几个功能发展成为一个庞大的协议簇。TCP/IP这种适应性使得IP地址的管理更加困难。本节仅介绍了IP地址管理的一些基本内容，在第6章中将会介绍一些更高级的技术。

1.3.1 首个八位组字节规则

如果不对网络作太过精确的划分，那么网络可以按照主机数量分为3类：大型网络、中型网络和小型网络。

- 大型网络 ——可以定义为包含大量主机的网络。大型网络的数量相对很少。
- 小型网络 ——作为大型网络的对照，它仅仅包含很少数量的主机，但小型网络的数目很多。
- 中型网络 ——相对于大型和小型网络来说，包含的主机数量中等，而且中型网络的数量也中等。

对于这3种规模的网络，高层的地址划分要求有3种类型的网络地址。面向大型网络的地址需要有能力和大量的主机编址，但是由于大型网络的数量有限，所以大型网络仅需要少量的网络地址。

而对小型网络来说情况又颠倒过来了，因为小型网络数量庞大，所以需要大量的小型网络的网络地址。但是小型网络主机有限，所以仅需要少量主机地址。

对于中等规模的网络来说，网络地址和主机地址的需求量均趋于中等水平。图1-5显示了3类IPv4地址的网络号和主机号是怎样划分的。

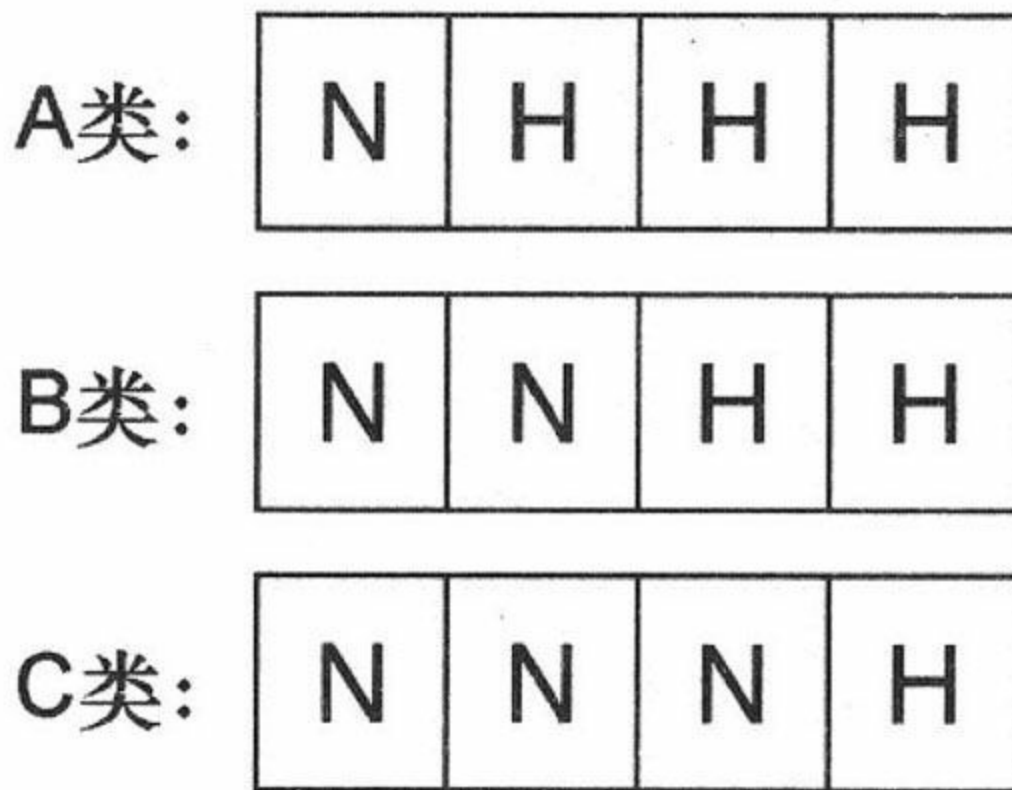


图1-5 A、B和C类IPv4地址格式

迄今为止，对于所描述的大型、中型和小型网络，是按照如下方式映射到各类地址的：

- **A类地址** ——用于大型网络，第1个八位组是网络号，后3个八位组是主机号。8位的网络号最多可以表示256个网络，而每个网络地址的主机号可以提供的主机数量为 2^{24} 或16777216。
- **B类地址** ——用于中型网络。前2个八位组表示网络号，后2个八位组表示主机号。网络号和主机号的数量均为 2^{16} 或65536个。
- **C类地址** ——对应于A类IP地址。前3个八位组表示网络号，最后1个八位组表示主机号。

因为所有的IPv4地址都是32位二进制字符串，所以需要某种方法来区分一个特定地址到底是属于哪一类地址。表1-3所示的首个八位组规则提

供了这种方法，如下所述：

- 对于A类地址，首个八位组的第1位，即32位字符串最左边的1位，总是被设置为0。因此，通过设置首个八位组的剩余位为0（最小）或为1（最大），我们可以找到A类地址范围中的最小数和最大数。于是我们可以得到最小数和最大数分别为0和127，但是这里有几个例外：0被保留作为缺省地址部分（参见第12章），127被保留为内部回送地址。[\[9\]](#)剩下的十进制数则是1~126。因此任何首个八位组落在1和126之间的IP地址均属于A类地址。

表1-3 首个八位组字节规则

规则	最小值与最大值	十进制范围
A类：第一位恒为0	00000000=0 01111111=127	1~126*
B类：第一、二位恒为10	10000000=128 10111111=191	128~191
C类：第一、二、三位恒为110	11000000=192 11011111=223	192~223

* 0和127保留。

- B类地址总是把左边的第1位设置为1，第2位设置为0。那么再次通过设置首个八位组的剩余位为0或为1，我们依然可以找到最小数和最大数。在图1-4中，我们可以看到首个八位组落在128和191之间的IP地址属于B类地址。
- 在C类地址中，前2位均被设置为1，第3位被设置为0。这样设置的结果是首个八位组在192和223之间。[\[10\]](#)

到目前为止，IPv4的编址看上去并不是十分困难。路由器和主机通过首个八位组字节规则能够很容易地确定IP地址的网络号。如果第1位是0，需要读取前8位才能获取网络地址；如果前两位是10，那么需要读取16位；如果前3位是110，则需求读取24位才能获取网络号。不幸的是，事情并不会这样简单。

1.3.2 地址掩码（Address Mask）

表示整个数据链路的地址——非特指某台主机的网络地址，可以用IP地址的网络部分来表示，其中主机位全部为0。例如，IP地址管理机构[11]可以将172.21.0.0[12]分配给一个申请者。因为172在128和191之间，所以这是一个B类地址，其中后两个八位组作为主机位，全部被设置为0。虽然前16位（172.21.）已经被指定，但是地址所有者有权决定后16位主机位的使用。

每一台设备和接口都将被分配一个惟一的、主机号明确的地址，例如172.21.35.17。不管设备是路由器还是主机，显然都需要知道自身的地址，而且它还需要能够确定它所属的网络，在这个案例中，它属于172.21.0.0。

这一任务通常由地址掩码来完成。地址掩码是一个32位的字符串，与IPv4地址的每一位相对应。掩码也可以像IPv4地址一样用点分十进制表示。这种表示方法会成为某些初学者的绊脚石。虽然地址掩码可以用点分十进制书写，但是它并不是一个地址。表1-4给出了对应于3类IPv4地址的标准地址掩码。

表1-4 A类、B类和C类IPv4地址的地址掩码

类	掩码	点分十进制表示
A	11111111000000000000000000000000	255.0.0.0
B	11111111111111110000000000000000	255.255.0.0
C	11111111111111111111111100000000	255.255.255.0

对于每一位IPv4地址位，设备会拿它与地址掩码的对应位进行布尔（逻辑）AND操作。AND函数表述如下：

比较两位并得出结果。当且仅当两位全部为1时，结果为1。如果两位中任意一位为0，则结果为0。

对于一个指定的IPv4地址，图1-6给出了怎样用地址掩码确定网络地址。地址掩码值为1的位对应于地址的网络位，值为0的位对应于主机位。因为172.21.35.17是B类地址，所以掩码前两个八位组必须全部设置为1，后两个八位组，即主机号的所有位必须设置为0。参见表1-4，这个掩码的点分十进制表示为255.255.0.0。

在IPv4地址和地址掩码的每一位上执行逻辑“与”（AND）操作，结果如图1-6所示。在结果中，网络位不变，所有主机位则变为0。通过向接口分配地址172.21.35.17和掩码255.255.0.0，设备将会知道接口属于网络172.21.0.0。对IPv4地址和掩码应用AND操作总能够得到网络地址。

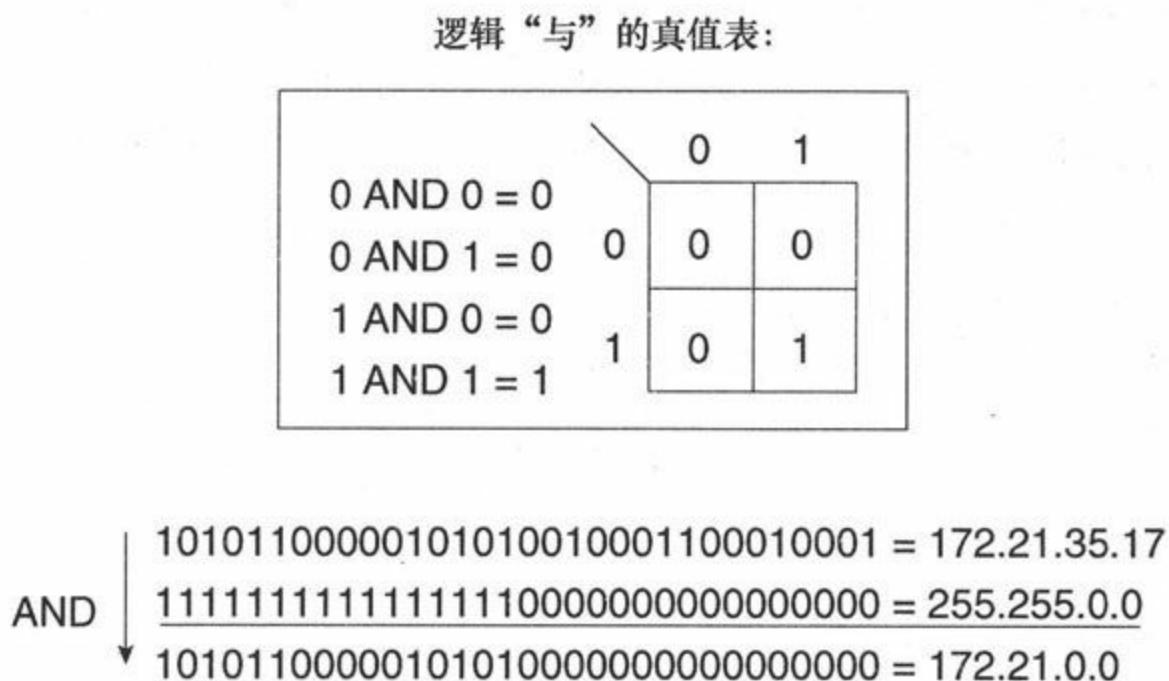


图1-6 B类地址的每一位与地址掩码的对应位进行AND操作，然后得到网络地址

通过下面命令可以向Cisco路由器的接口分配地址和掩码（本例中接口为E0）：

```
Smokey(config)# interface ethernet 0
Smokey(config-if)# ip address 172.21.35.17 255.255.0.0
```

但是为什么要使用地址掩码？到目前为止，使用首个八位组字节规则看上去更简单一些。

1.3.3 子网和子网掩码

首先，决不要忽略网络层地址的必要性。为了完成路由选择，每个数据链路（网络）都必须有一个惟一的地址；另外，数据链路上的每台主机也必须有一个地址，这个地址不仅标识主机为一个网络成员，还可以把主机与网络上的其他主机区分开来。

到目前为止的定义中，一个A类、B类或C类地址仅仅能用于一个单一网络中；为了建立一个网络，每个数据链路都必须使用不同的地址，以便这些网络可以被惟一地标识。如果每一个数据链路都使用一个单独的A类、B类或C类地址，那么即使用尽所有的IPv4地址，也只能给少于1700万个数据链路分配地址。显然，这种方法是不切实际的，[\[13\]](#)在上面的例子中，如果充分地使用主机地址空间，那么在数据链路172.21.0.0中的设备数目可以超过65000！

使A类、B类或C类地址实用化的惟一方法是对主网地址进行划分，例如将172.21.0.0划分为子网地址。请回忆两个事实：

- IPv4地址的主机部分可以随意使用。
- IPv4地址的网络号由分配给接口的地址掩码确定。

如图1-7所示，分配给网络的地址为B类地址172.21.0.0。5个数据链路将主机和路由器互连起来，每个数据链路都需要一个网络地址。按照目前的情况，172.21.0.0必须分配给其中的一个数据链路，那么另外4个数据链路还需要4个地址。

注意图1-7所示，地址掩码并不是标准的16位B类地址掩码；而是被扩展了8位，以便IP地址的前24位都被解释为网络位。换句话说，掩码使路由器和主机把读取的前8位主机位作为网络地址的一部分。结果是，主网络地址应用于整个网络，而每一个数据链路则变为一个子网（subnet）；一个子网是一个主A类、B类或C类地址空间的一个子集。

现在，IPv4地址包括3个部分：网络部分、子网部分和主机部分。地址掩码现在变为子网掩码，或比标准地址掩码长的掩码。地址的前两个八位组依然是172.21，但是第3个八位组——主机位已经由子网位代替一的变化范围为0~255。在图1-6中的网络有子网1、2、3、4和5（172.21.1.0~172.21.5.0）。在单一B类地址下最多可以有256个子网，对应的掩码如图1-7所示。

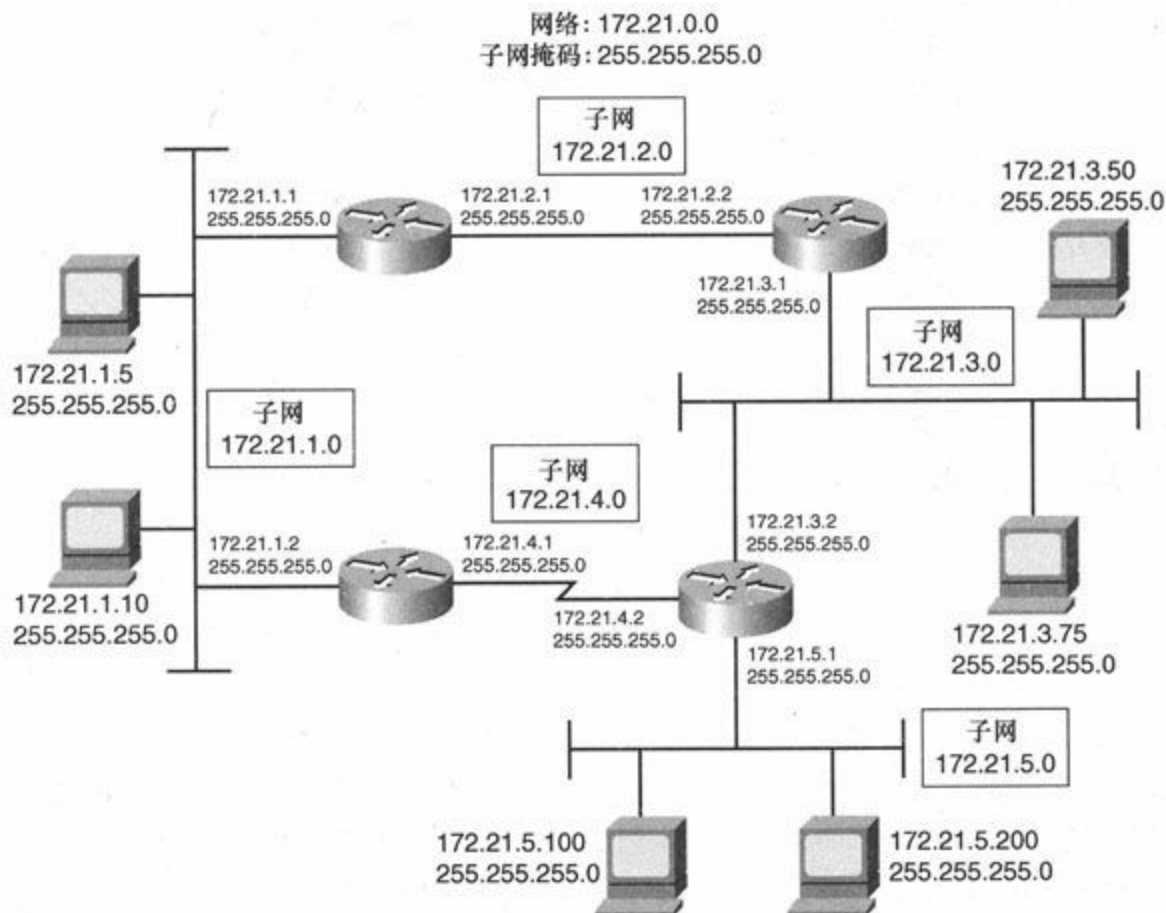


图1-7 通过向主机位借位用作子网位，子网掩码使一个单一的网络地址可以用于多个数据链路

下面给出两点告诫。首先，并不是所有路由选择协议都支持子网地址，即子网位全0或全1。因为这些协议是有类别化协议，它们不能区分一个全0子网和主网络号。例如，在图1-7中子网0为172.21.0.0；而主网IP地址也为172.21.0.0。没有更多信息将无法区分二者。

同样的，有类别路由选择协议也不能区分全1子网的广播地址和一个所有子网的广播地址。[\[14\]](#)例如，图1-7中的全1子网为172.21.255.0。对于这个子网，广播地址是172.21.255.255，但是这也是在主网172.21.0.0的所有子网上所有主机的广播地址。没有更多的信息也无法区分二者。第1版RIP协议和IGRP协议都是有类别路由选择协议；第7章将会介绍无类别路由选择协议，这种路由选择协议才可以真正地使用全0或全1子网。

其次是与子网及其掩码的口头表述有关。在图1-7中，对B类地址的第3

个八位组进行子网划分是非常普遍的，但还常常听到人们这样表述子网设计：“B类地址使用C类地址掩码”，或者“将B类地址划分为C类地址”。这两种表述都是错误的。它们常常会对子网设计引起误解或者是不准确的理解。对于图1-6中所示的子网划分图解的正确表述应该是“一个使用8位进行子网划分的B类地址”或者“一个带有24位掩码的B类地址”。

可以用以下3种格式中的任何一种表示子网掩码：

点分十进制：255.255.255.0

位计数：172.21.0.0/24

十六进制：0xFFFFF00

虽然位计数格式变得渐渐流行起来，但是点分十进制暂时一段时期仍旧经常使用在一些软件里面。与点分十进制相比，位计数格式更容易书写（地址后面是/，/后面紧跟着是网络部分的位计数）。另外，位计数格式可以更清楚地描述掩码的实际作用，因而可以避免前面段落出现的语义误解问题。某些UNIX系统使用十六进制格式。

虽然在Cisco路由器中必须使用点分十进制方式表示地址掩码，但是在行配置模式下使用命令**ip netmask-format[decimal|hexadecimal|bit-count]**，可以设置使用3种格式中的任何一种格式显示掩码。例如，为使路由器以位计数格式显示掩码，配置如下：

```
Gladys(config) # line vty 0 4
```

```
Gladys(config-line)# ip netmask-format bit-count
```

1.3.4 子网规划

如前面部分所述，在有类别地址环境中，子网位不能全部为0或全部为1。同样的，一个主机的IPv4地址也不能将主机位全部设置为0，这种用法是为路由器保留的，用于表示网络和子网自身。当然IPv4地址的主机

位也不能全部被设置为1，因为它用于表示广播地址。所有这些限制无一例外地适用于IP地址的主机位，并且这也是子网规划的起点。除了这些限制，网络设计人员还需要根据地址空间与网络详细的匹配程度来选择最合理的子网划分方案。

在规划子网和子网掩码时，可以使用相同的公式计算一个主网地址下可用的子网数以及每个子网内可用的主机数，公式为： $2^n - 2$ ，其中 n 表示子网位数或主机空间，2表示减去全0和全1两个不可用地址。例如，给定一个A类地址10.0.0.0，子网掩码10.0.0.0/16（255.255.0.0）意味着有8位子网空间，也就是可以产生 $2^8 - 2 = 254$ 个子网，每个子网可以有 $2^{16} - 2 = 65534$ 个主机地址。另一方面，掩码10.0.0.0/24（255.255.255.0）表示有16位子网空间，可以产生65534个子网，其中8位主机空间可以在某个子网中产生254个主机地址。

下面是IPv4地址子网划分的步骤：

步骤1： 确定需要多少个子网，每个子网需要多少台主机。

步骤2： 为了满足第1步提出的需求，使用公式 $2^n - 2$ 确定子网位数和主机位数。如果存在多个子网掩码可以满足第1步需求，那么选择最能够符合未来需求的一个。例如，如果网络最有可能通过增加子网发展起来，那么选择子网位最多的掩码；如果网络最有可能借助增加现有子网内的主机数发展起来，则选择主机位最多的掩码。为了避免所选择的方案中的子网及子网内的主机地址被迅速地用完，需要为将来的发展预留一些空间。

步骤3： 使用二进制进行计算，在子网空间中确定所有的位组合方式。在每种组合方式中，将所有主机位都设置为0，将得到的子网地址转换为点分十进制格式。最终结果就是子网地址。

步骤4： 对于每一个子网地址，再次使用二进制，在保持子网位不变的情况下写出所有主机位组合，并将结果转换成点分十进制格式。最终结果就是每个子网的可用主机地址。

这里没有过分强调在最后两步中使用二进制的重要性。当进行子网划分时，最主要的惟一错误根源就是，在没有理解在二进制上会发生什么的情况下试图使用点分十进制方法。此外，点分十进制表示法对于人们读写IPv4地址十分方便。但是路由器和主机却把地址看作32位二进制字符

串；为了顺利地完地址操作，必须采用路由器和主机处理地址的方式。

就目前给出的例子而言，作者在前面的段落中似乎有点多虑了。在没有限定必须使用二进制方式表示地址和掩码的时候，子网模式和主机地址看上去还是十分清楚的。下面小节将讨论使用4个步骤完成子网规划，在那里点分十进制表示法将十分不明确。

1.3.5 打破八位组界线

到目前为止，在给出的例子中，子网空间都是以八位组为界线的。但这并不总是最实用或最有效的选择。例如，如果你需要对一个B类地址进行子网划分，并满足以下需求：数据链路数为500，每个数据链路内主机数不超过100台，应该怎么办？这样的需求很容易得以满足，只要使用9位子网位，就可以得到 $2^9 - 2 = 510$ 个子网，剩下7位做主机位，每个子网的可用主机数为 $2^7 - 2 = 126$ 。除此不再有其他位组合可以满足上面的需求。

请注意，如果还是以八位组为界线的话，那么将无法对C类地址进行子网划分。如果要这样做就会占用最后1个八位组，那么就没有更多主机位了。因此，如下面的例子所示，子网位和主机位必须共享最后1个八位组。

图1-8与图1-7中显示的网络除了分配的地址是C类地址192.168.100.0之外，其他完全相同。

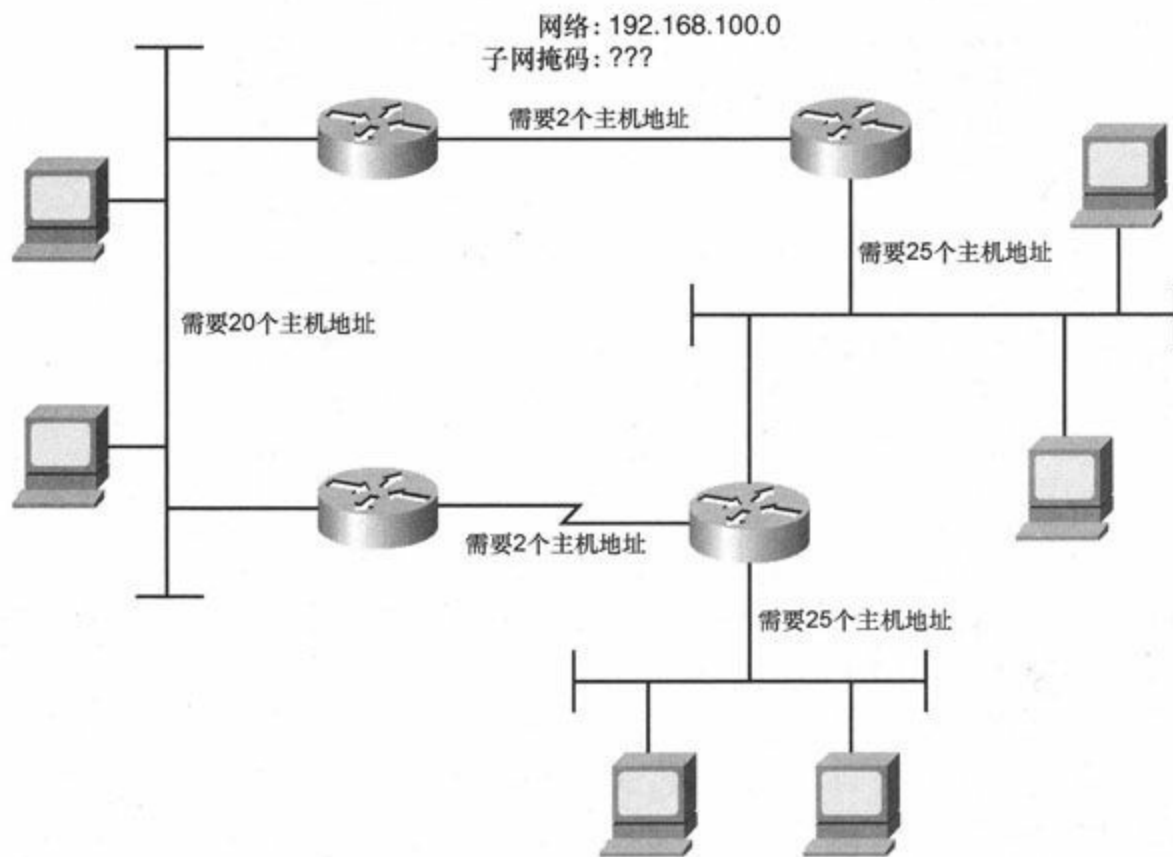


图1-8 除了分配C类地址之外，这里沿用图1-7给出的网络。如果子网位占用整个八位组，那么将无法进行划分，因为主机位将没有空间

在这个网络中共有5条数据链路，因此至少需要划分出5个子网地址。图中还指明了每个子网需要分配的主机数（包括路由器接口）。其中两个以太网最多需要25个主机地址。所以完整的子网划分最小需求是5个子网，每个子网至少需要25个主机地址。

使用公式 $2^n - 2$ 可以计算出，3个子网位和5个主机位即可以满足需求： $2^3 - 2 = 6$ ， $2^5 - 2 = 30$ 。带有3位子网位的C类地址掩码可以用点分十进制表示为255.255.255.224。

图1-9给出了子网位的推导过程。用二进制表示第2步计算出的子网掩码，子网掩码下面是IP地址。垂直线标记了子网空间，从二进制0开始计数，在这一空间中的所有位组合均被写出。

$$11000000101010000110010000000000 = 192.168.100.0$$

网络	000	主机
地址	001	地址
空间	010	空间
	011	
	100	子网
	101	地址
	110	空间
	111	

在图2-10中，不发生变化的网络位填写在子网空间的左边，全部为0的主机位填写在子网位的右边。结果被转换为点分十进制表示后，得到6个子网地址（记住，第一个和最后一个地址，即在子网空间中全部位为0和全部位为1的地址不能使用）。

后一个地址主机位全部为1，这是子网192.168.100.32的广播地址。主机地址从子网地址起到广播地址为止。按照顺序，下一个子网地址是192.168.100.64。

现在，在二进制层次上理解子网划分的重要性就显而易见了。给出一个地址，如192.168.100.160，你不能确定它是否是一个主机地址、子网地址或广播地址。甚至在子网掩码已知情况下，结论也并不总是明显的。

这里我们鼓励读者计算例子中所有余下子网的主机地址，并且仔细观察产生地址的模式，理解这些模式对下一部分的内容会有帮助。

网络位	主机位	
110000001010100001100100001	00000	= 192.168.100.32 ← 子网号
110000001010100001100100001	00001	= 192.168.100.33
110000001010100001100100001	00010	= 192.168.100.34
110000001010100001100100001	00011	= 192.168.100.35
110000001010100001100100001	00100	= 192.168.100.36
110000001010100001100100001	00101	= 192.168.100.37
110000001010100001100100001	00110	= 192.168.100.38
110000001010100001100100001	00111	= 192.168.100.39
110000001010100001100100001	01000	= 192.168.100.40
110000001010100001100100001	01001	= 192.168.100.41
110000001010100001100100001	01010	= 192.168.100.42
110000001010100001100100001	01011	= 192.168.100.43
110000001010100001100100001	01100	= 192.168.100.44
110000001010100001100100001	01101	= 192.168.100.45
110000001010100001100100001	01110	= 192.168.100.46
110000001010100001100100001	01111	= 192.168.100.47
110000001010100001100100001	10000	= 192.168.100.48
110000001010100001100100001	10001	= 192.168.100.49
110000001010100001100100001	10010	= 192.168.100.50
110000001010100001100100001	10011	= 192.168.100.51
110000001010100001100100001	10100	= 192.168.100.52
110000001010100001100100001	10101	= 192.168.100.53
110000001010100001100100001	10110	= 192.168.100.54
110000001010100001100100001	10111	= 192.168.100.55
110000001010100001100100001	11000	= 192.168.100.56
110000001010100001100100001	11001	= 192.168.100.57
110000001010100001100100001	11010	= 192.168.100.58
110000001010100001100100001	11011	= 192.168.100.59
110000001010100001100100001	11100	= 192.168.100.60
110000001010100001100100001	11101	= 192.168.100.61
110000001010100001100100001	11110	= 192.168.100.62
110000001010100001100100001	11111	= 192.168.100.63 ← 广播地址

— 有效主机地址

图1-11 写出主机空间中所有位组合可以得到子网内的主机地址。这里是子网192.168.100.32的主机位

1.3.6 子网掩码的故障诊断

在“解剖”一个给定的主机地址和掩码时，常常需要确定地址属于哪个子网。例如，如果在一个接口上配置了地址，一个很好的实践就是首先验证对于接口连接的子网来说该地址是否合法。

使用下面的步骤逆推一个IP地址：

步骤1： 用二进制写下一个给定的子网掩码。

步骤2： 用二进制写下一个主机IPv4地址。

步骤3： 在知道一个地址的类别后，掩码的子网位便是显然的了。根据掩码位，在最后网络位和第1个子网位之间画一条线，在最后子网位和第1台主机之间也画另一条线。

步骤4： 写下地址的网络位和子网位，设置所有的主机位为0。最终的结果就是主机地址所属的子网地址。

步骤5： 再次写下地址的网络位和子网位，这次设置所有主机位为1。结果就是本子网的广播地址。

步骤6： 按照顺序可以知道第一个地址是子网地址，最后一个地址是广播地址。而且还

可以知道在这两个地址之间的所有地址都是合法的主机地址。

对于地址172.30.141/25，图1-12给出了以上步骤的示例。

这个地址是B类地址，所以前16位是网络位，25位掩码中的后9位是子网位。可以发现子网地址是172.30.0.128，广播地址是172.30.0.255。在这两个地址之间的主机地址对于这个子网来说都是合法的，如对子网172.30.0.128来说，172.30.0.129~172.30.0.254都是主机地址。

172.30.0.141/25

(1) 写出子网掩码:	11111111111111111111111100000000 = 255.255.255.128
(2) 写出IP地址:	101011000000111100000000010001101 = 172.30.0.141

(3) 标记子网空间。	11111111111111111111111100000000 = 255.255.255.128
	101011000000111100000000010001101 = 172.30.0.141

导出...	11111111111111111111111100000000 = 255.255.255.128
	101011000000111100000000010001101 = 172.30.0.141
(4) 子网地址:	101011000000111100000000010000000 = 172.30.0.128
(5) 广播地址:	101011000000111100000000011111111 = 172.30.0.255

图1-12 给定一个IPv4地址和子网掩码，按照以上这些步骤可以找出子网地址、广播地址和主机地址

在这个例子中，初次进行子网划分的人可能会受到以下几种情况的干扰。一种是地址的第3个八位组所有位都为0。另一种是最后一个八位组仅一个子网位。一些人可能会认为广播地址看上去不合法。所有这些不舒服的感觉都源自地址的点分十进制表示法。当使用二进制表示地址和掩码时，这些疑虑会被打消，任何事看上去都一切正常，掩码设定了9位子网空间——包括第3个八位组和第4个八位组的第1位。这个案例说明了如果使用二进制表示法时一切正常，那么就不必担心看上去有些奇怪的点分十进制表示法。

1.4 地址解析协议（ARP）

第1章解释了通过读取和操作数据包的网络地址，路由器可以沿逻辑路径传送数据包，其中逻辑路径包括多个数据链路。沿独立的数据链路传送数据包时，需要把数据包封装在帧中，并且使用数据链路标识（如MAC地址）让帧可以从链路的源点到达目的地。本书的主题之一是为了进行路由选择，路由器利用何种机制发现并共享地址信息。类似的，数据链路上的设备也需要一种方法发现邻居的数据链路标识，以便将数据帧传送到正确的目的地。

有几种机制可以提供这些信息；[\[15\]](#) IPv4使用地址解析协议（ARP），详见RFC826。图1-13给出了ARP的工作机制。当一台设备需要发现另一台设备的数据链路标识符时，它将建立一个ARP请求数据包。这个请求数据包中包括目标设备的IPv4地址以及请求设备（发送者）的源点IPv4地址和数据链路标识符（MAC地址）。然后ARP请求数据包被封装在数据帧中，其中带有作为源的发送者的MAC地址和作为目标的广播地址（参见示例1-6）。[\[16\]](#)

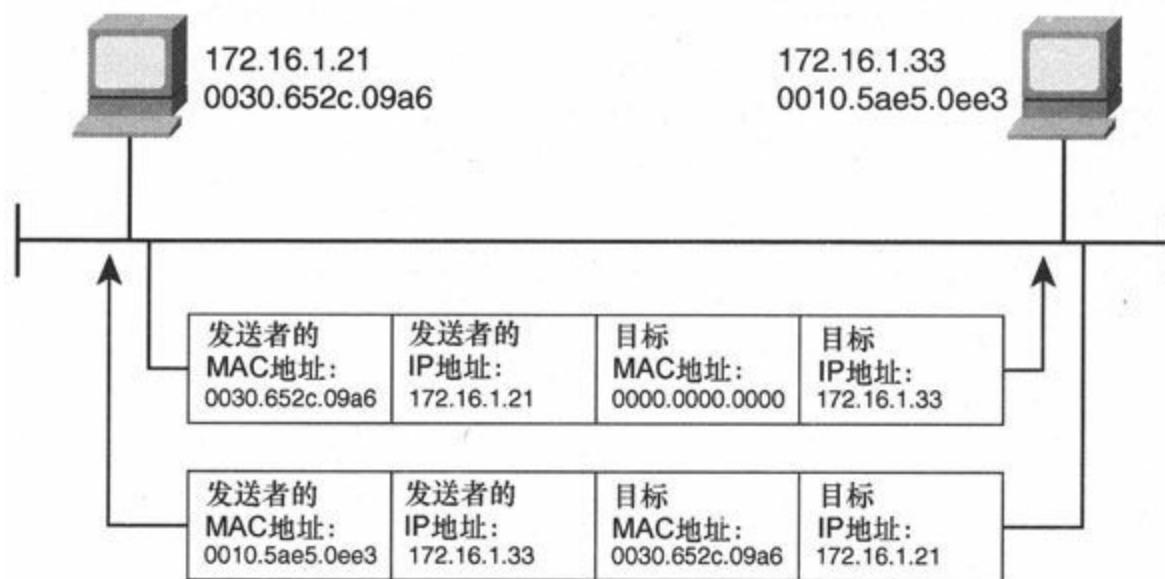


图1-13 ARP用于把设备的数据链路标识符映射到它的IP地址上

示例1-6 协议分析器捕捉到图1-13所描述的ARP请求数据包及封装帧

```
Ethernet II, Src: 00:30:65:2c:09:a6, Dst: ff:ff:ff:ff:ff:ff
  Destination: ff:ff:ff:ff:ff:ff (Broadcast)
  Source: 00:30:65:2c:09:a6 (AppleCom_2c:09:a6)
  Type: ARP (0x0806)
Address Resolution Protocol (request)
  Hardware type: Ethernet (0x0001)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (0x0001)
  Sender MAC address: 00:30:65:2c:09:a6 (AppleCom_2c:09:a6)
  Sender IP address: 172.16.1.21 (172.16.1.21)
  Target MAC address: 00:00:00:00:00:00 (00:00:00_00:00:00)
  Target IP address: 172.16.1.33 (172.16.1.33)
```

广播地址意味着数据链路上的所有设备都将收到该帧，并且要检查帧内封装的数据包。除了目标机可以识别此数据包外，其他所有设备都会丢弃此数据包。目标机将向源地址发送ARP响应数据包，提供它的MAC地址（参见示例1-7）。

示例1-7 协议分析器捕捉的图1-13所描述的ARP响应数据包

```
Ethernet II, Src: 00:10:5a:e5:0e:e3, Dst: 00:30:65:2c:09:a6
  Destination: 00:30:65:2c:09:a6 (AppleCom_2c:09:a6)
  Source: 00:10:5a:e5:0e:e3 (3com_e5:0e:e3)
  Type: ARP (0x0806)
  Trailer: 15151515151515151515151515151515...
Address Resolution Protocol (reply)
  Hardware type: Ethernet (0x0001)
  Protocol type: IP (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (0x0002)
  Sender MAC address: 00:10:5a:e5:0e:e3 (3com_e5:0e:e3)
  Sender IP address: 172.16.1.33 (172.16.1.33)
  Target MAC address: 00:30:65:2c:09:a6 (AppleCom_2c:09:a6)
  Target IP address: 172.16.1.21 (172.16.1.21)
```

当调用调试功能**debug arp**时，Cisco路由器可以显示ARP的活动情况，参见示例1-8。

示例1-8 路由器**Aretha**（**172.21.5.1**）响应来自主机**172.19.35.2**的**ARP**请求

```
Aretha#debug arp
IP ARP: rcvd req src 172.19.35.2 0002.6779.0f4c, dst 172.21.5.1 Ethernet0
IP ARP: sent rep src 172.21.5.1 0000.0c0a.2aa9,
          dst 172.19.35.2 0002.6779.0f4c Ethernet0
Aretha#
```

图1-14给出了ARP数据包的格式。这里可以把图中描述的各字段同示例1-6和示例1-7的ARP数据包相对照。

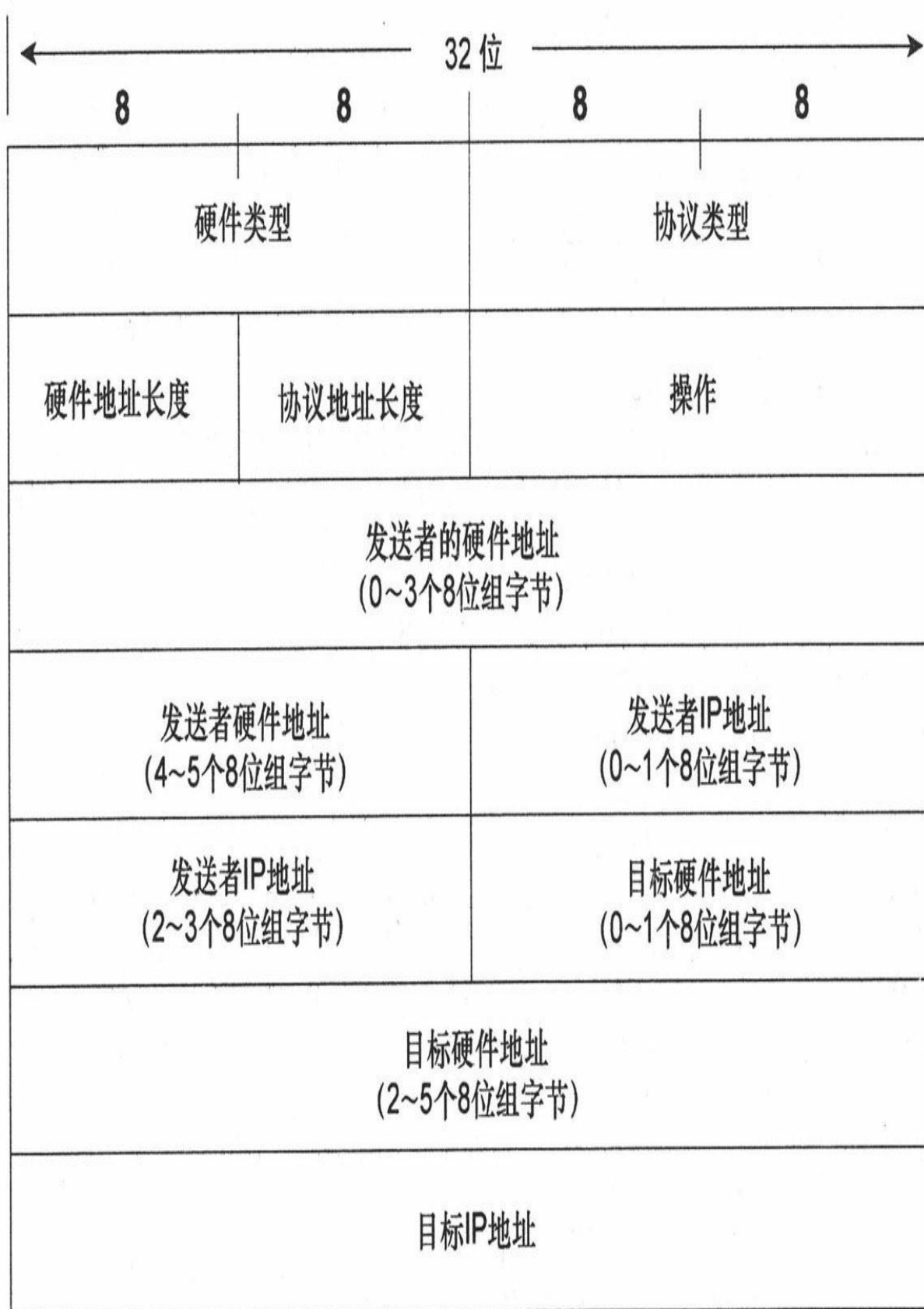


图1-14 ARP数据包格式

- 硬件类型（**Hardware Type**）——指定了硬件的类型，详见IETF的规范说明。[\[17\]](#)一些常用的类型编号如表1-5所示。

表1-5 常用的硬件类型码

编号	硬件类型
1	以太网
3	X.25
4	Proteon ProNET Token Ring
6	IEEE 802 网络
7	ARCnet
11	Apple LocalTalk
14	SMDS
15	帧中继
16	异步传输模式（ATM）
17	高速数据链路控制（HDLC）
18	光纤信道
19	异步传输模式（ATM）
20	串行链路

- 协议类型（**Protocol Type**）——指定了发送者映射到数据链路标识符的网络层协议的类型；IP对应0x0800。
- 硬件地址长度（**Hardware Address Length**）——指定了数据链路标识符的长度，单位是八位组。MAC地址的长度为6。
- 协议地址长度（**Protocol Address Length**）——指定了网络层地址的长度，单位是八位组。IPv4地址的长度为4。
- 操作（**Operation**）——指明了一个数据包是ARP请求（1）还是ARP响应（2）。

这里还可以发现有其他的值表明ARP数据包的其他用途。如反向ARP请

求（3）、

反向ARP响应（4）、反转ARP请求（8）、反转ARP响应（9）。

最后20个八位组是发送者和目标机的数据链路标识符和IPv4地址。

在示例1-9所示屏幕的最上面，命令**show arp** 用于检查Cisco路由器内的ARP表。

示例1-9 连接到相同网络上的3台设备的ARP表：**Cisco**路由器，**Windows**主机和**Linux**主机

```
Martha#show arp
```

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	10.158.43.34	2	0002.6779.0f4c	ARPA	Ethernet0
Internet	10.158.43.1	-	0000.0c0a.2aa9	ARPA	Ethernet0
Internet	10.158.43.25	18	00a0.24a8.a1a5	ARPA	Ethernet0
Internet	10.158.43.100	6	0000.0c0a.2c51	ARPA	Ethernet0

```
Martha#
```

```
C:\WINDOWS>arp -a
```

Interface: 148.158.43.25

Internet Address	Physical Address	Type
10.158.43.1	00-00-0c-0a-2a-a9	dynamic
10.158.43.34	00-02-67-79-0f-4c	dynamic
10.158.43.100	00-00-0c-0a-2c-51	dynamic

```
Linux:~# arp -a
```

Address	HW type	HW address	Flags	Mask
10.158.43.1	10Mbps Ethernet	00:00:0C:0A:2A:A9	C	*
10.158.43.100	10Mbps Ethernet	00:00:0C:0A:2C:51	C	*
10.158.43.25	10Mbps Ethernet	00:A0:24:A8:A1:A5	C	*

```
Linux:~#
```

请注意年龄一栏，这一栏表明为了防止陈旧信息充满ARP表，每经过一个特定的实际间隔，ARP信息将会被刷新。Cisco路由器保存ARP表项的时间为4个小时（14400s）；这个缺省值可以修改。下面的例子就是将ARP的超时值修改为30min（1800s）：

```
Martha (config) # interface Ethernet 0
```

```
Martha (config-if) # arp timeout 1800
```

示例1-9所示屏幕的中间给出了Windows PC的ARP表，屏幕底部给出了Linux机器的ARP表。虽然它们的格式不同于Cisco路由器的ARP表，但是3个表中的实质性信息是相同的。

ARP表项还可以永久地保存在表中。为了实现地址172.21.5.131到硬件地址0000.00a4.b74c的静态映射，并且采用SNAP（Subnetwork Access Protocol）封装类型，可以使用以下命令完成：

```
Martha(config) # arp 172.21.5.1310000.00a4.b74c snap
```

命令**clear arp-cache**可以从ARP表中强制删除所有动态表项。并且此命令也可以清除快速交换高速缓冲区和IP路由高速缓冲区中的内容。

ARP还有几种变形，其中至少有一种对路由选择十分重要，它就是代理ARP。

1.4.1 代理ARP

代理ARP有时也被叫做混杂ARP，详见RFC925和RFC1027，代理ARP被路由器作为向主机表明自身可用的一种手段。例如，主机192.168.12.5/24需要向主机192.168.20.101/24发送数据包，但是它没有配置缺省网关信息，因而也就不知道如何到达路由器。这时它可以向192.168.20.101发送一个ARP请求；本地路由器收到这一请求，并且路由器知道如何到达网络192.168.20.0，因此路由器将回复以上请求，其中把自己的数据链路标识符作为ARP回复数据包中的硬件地址。事实上，路由器欺骗了本地的主机，让它认为路由器的接口就是192.168.20.101的接口。最终所有发向192.168.20.101的数据包都被送往路由器。

图1-15给出了代理ARP的另一种用途。这里特别关注的是地址掩码。路由器配置的掩码是28位掩码（4个子网位的C类地址），而主机配置的是标准的C类地址掩码（24位）。其结果是主机并不知道子网的存在。当

主机192.168.20.66想发送数据包到192.168.20.25时，它首先将发送ARP请求。这时路由器识别出数据包的目标地址属于另一个子网，因而向请求主机回复自己的硬件地址。这种代理ARP使得子网化网络拓扑结构对主机来说是透明的。

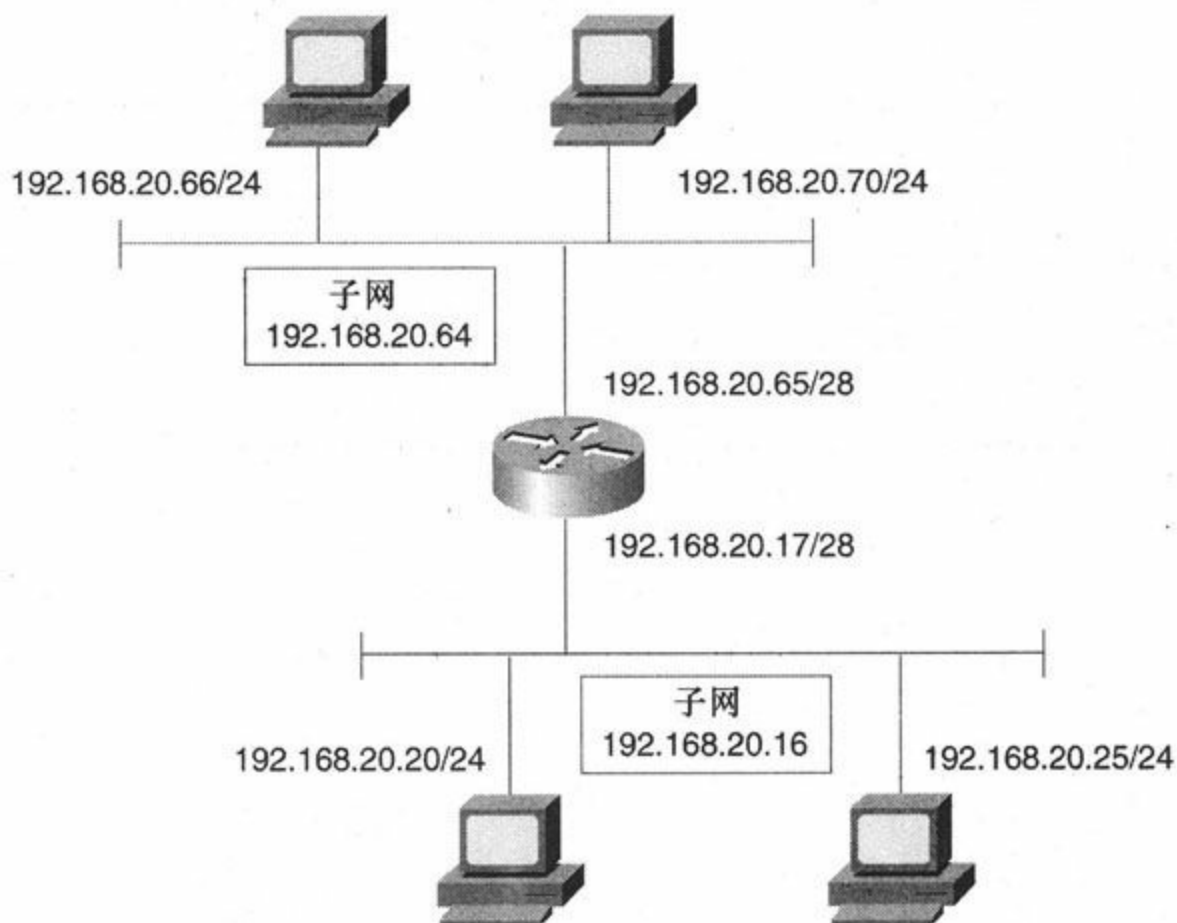


图1-15 代理ARP实现了子网划分的透明性

示例1-10所示的ARP高速缓冲暗示了代理ARP的又一用途。注意，有多个IPv4地址映射到单一的MAC标识符；其中IP地址对应着主机，而硬件MAC标识符属于路由器接口。

示例1-10 图1-15中主机192.168.20.66的ARP表显示出多个IPv4地址映射到单一MAC标识符，这说明正在使用代理ARP

```
C:\WINDOWS>arp -a

Interface: 192.168.20.66

Internet Address      Physical Address      Type
192.168.20.17         00-00-0c-0a-2a-a9     dynamic
192.168.20.20         00-00-0c-0a-2a-a9     dynamic
192.168.20.25         00-00-0c-0a-2a-a9     dynamic
192.168.20.65         00-00-0c-0a-2c-51     dynamic
192.168.20.70         00-02-67-79-0f-4c     dynamic
```

在IOS系统中，缺省情况下代理ARP功能是打开的，当然也可以在每个接口上使用命令**no ip proxy-arp** 关闭此功能。

1.4.2 无故ARP

主机偶尔也会使用自己的IPv4地址作为目标地址发送ARP请求。这种ARP请求称为无故ARP，主要有两个用途：

- 无故ARP可以用于检查重复地址。一台设备可以向自己的IPv4地址发送ARP请求，如果收到ARP响应则表明存在重复地址。
- 无故ARP还可以用于通告一个新的数据链路标识符。当一台设备收到一个ARP请求，如果ARP高速缓冲中已有发送者的IPv4地址，那么与此IPv4地址相对应的硬件地址将会被发送者新的硬件地址所更新。这种无故ARP用途正是基于此事实。
- 某个子网内运行热备份路由器协议（HSRP协议）的路由器如果从其他路由器变成了主路由器，它就会发出一个无故ARP来更新该子网上主机的ARP缓存。

许多IP实现中都没有实现无故ARP功能，但是读者应该知道它的存在。在IOS系统中缺省情况下是关闭的，但可以通过命令**ip gratuitous-arps** 激活它。

1.4.3 反向ARP

代替映射硬件地址到已知IPv4地址，反向ARP（RARP）可以实现IPv4地址到已知硬件地址的映射。某些设备，如无盘工作站在启动时可能不知道自己启动时的IPv4地址。嵌入这些设备固件中的RARP程序可以允许它们发送ARP请求，其中硬件地址为设备的硬件编入地址。RARP服务器将会向这些设备回复相应的IPv4地址。

RARP在很大程度上正在被动态主机配置协议（DHCP）和自举协议（BOOTP）的扩展协议所替代，不同于RARP，这两种协议都可以提供IPv4地址以外的更多信息，而且还可以跨越本地数据链路。

1.5 ICMP

Internet消息控制协议（ICMP）指定了多种消息类型，这些消息的共同目的就是管理网络，详见RFC792。ICMP的消息可以分为错误消息、请求消息和响应消息。图1-16给出了一般的ICMP数据包格式。数据包可以通过类型来标识，许多数据包类型都有多个指定的类型，可以用代码字段来标识它们。表1-6列出了多种ICMP的数据包类型和代码，详见RFC1700。

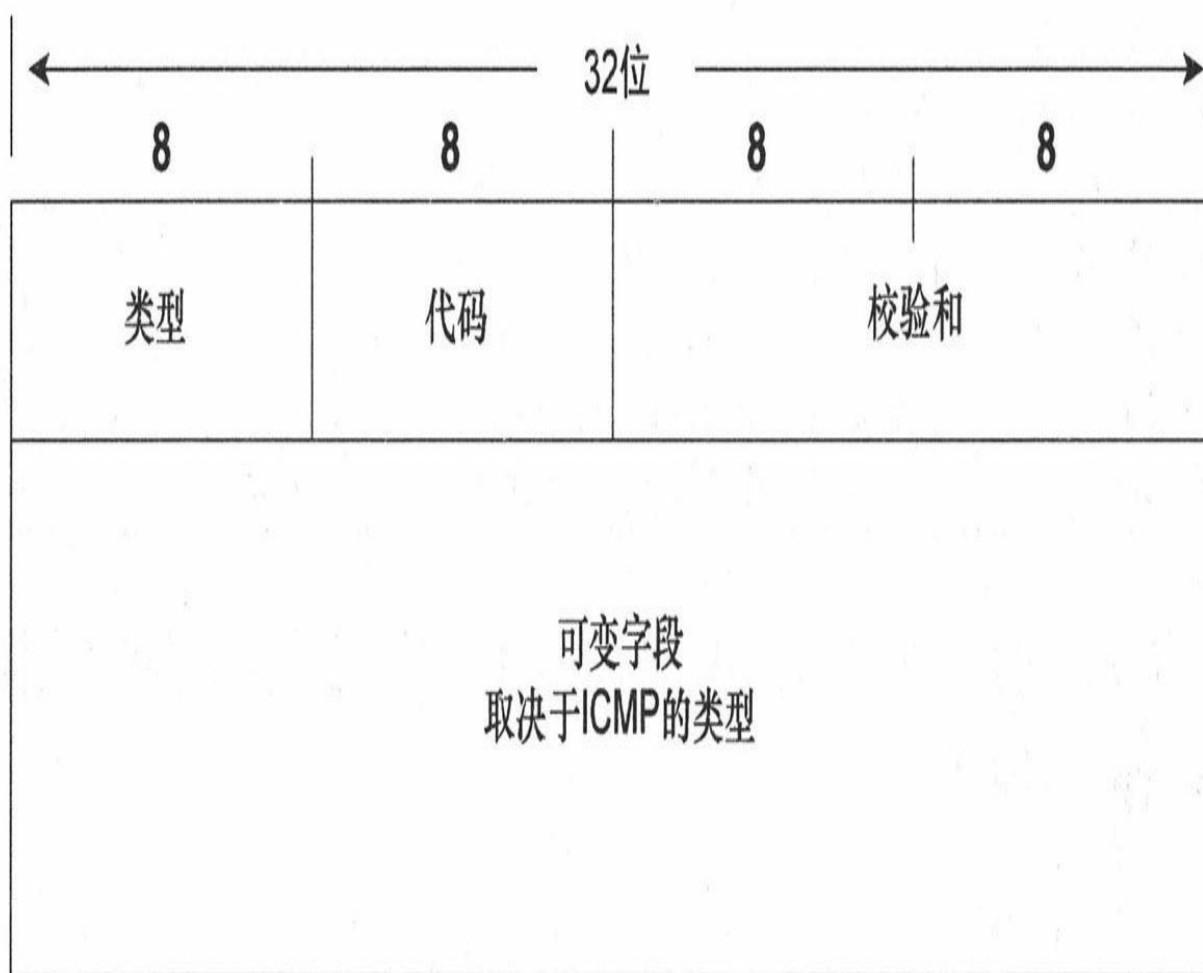


图1-16 ICMP数据包头包括类型字段，进一步标识某些类型的代码字段和校验和。剩余的字段依赖于特定类型和代码

表1-6 ICMP数据包类型字段和代码字段

类型	代码	名称
0	0	回应应答
		目的地不可达
	0	网络不可达
	1	主机不可达
	2	协议不可达
	3	端口不可达
	4	需要分段和不需要分段标记置位
	5	源路由失败
	6	目的网络未知
	7	目的主机未知
	8	源主机被隔离
3	9	与目的网络的通信被禁止
	10	目的主机的通信被禁止
	11	对请求的服务类型，目的网络不可达
	12	对请求的服务类型，目的主机不可达
4	0	源抑制（Source Quench）
5		重定向
	0	为网络（子网）重定向数据报
	1	为主机重定向数据报
	2	为网络和服务类型重定向数据报
6	3	为主机和服务类型重定向数据报
	0	选择主机地址
	0	回应
8	0	路由器通告
9	0	路由器选择
10	0	超时

11	0	传输中超出TTL
	1	超出分段重组时间
		参数问题
12	0	指定错误的指针
	1	缺少需要的选项
	2	错误长度
13	0	时间戳
14	0	时间戳回复
15	0	信息请求（废弃）
16	0	信息回复（废弃）
17	0	地址掩码请求（即将废弃）
18	0	地址掩码回复（即将废弃）
30	-	路由跟踪

示例1-11和示例1-12给出了协议分析器捕捉到的两种众所周知的ICMP消息——Echo请求和Echo回复，它们常用在ping命令的功能中。

示例1-11 ICMP的Echo消息及其IPv4头部

Internet Protocol, Src Addr: 172.16.1.21 (172.16.1.21),

Dst Addr: 198.133.219.25 (198.133.219.25)

Version: 4

Header length: 20 bytes

Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)

Total Length: 84

Identification: 0xabc3 (43971)

Flags: 0x00

Fragment offset: 0

Time to live: 64

Protocol: ICMP (0x01)

Header checksum: 0x8021 (correct)

Source: 172.16.1.21 (172.16.1.21)

Destination: 198.133.219.25 (198.133.219.25)

Internet Control Message Protocol

Type: 8 (Echo (ping) request)

Code: 0

Checksum: 0xa297 (correct)

Identifier: 0x0a40

Sequence number: 0x0000

Data (56 bytes)

0000	40 fd ab c2 00 0e 73 57 08 09 0a 0b 0c 0d 0e 0f	@.....sW.....
0010	10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
0020	20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f	!"#\$%&'()*+,-./
0030	30 31 32 33 34 35 36 37	01234567

示例1-12 ICMP的Echo回复消息

Internet Protocol, Src Addr: 198.133.219.25 (198.133.219.25),
Dst Addr: 172.16.1.21 (172.16.1.21)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 84
Identification: 0xabc3 (43971)
Flags: 0x00
Fragment offset: 0
Time to live: 242
Protocol: ICMP (0x01)
Header checksum: 0xce20 (correct)
Source: 198.133.219.25 (198.133.219.25)
Destination: 172.16.1.21 (172.16.1.21)

Internet Control Message Protocol

Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0xaa97 (correct)
Identifier: 0x0a40
Sequence number: 0x0000
Data (56 bytes)

0000	40 fd ab c2 00 0e 73 57 08 09 0a 0b 0c 0d 0e 0f	@.....SW.....
0010	10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
0020	20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f	!"#\$%&'()*+,-./
0030	30 31 32 33 34 35 36 37	01234567

虽然大部分**ICMP** 类型都与路由选择功能有关，但是有3个类型特别重要：

- 路由器通告（**Router Advertisement**）和路由器选择（**Router Selection**）——分别是类型9和类型10，它们用于ICMP路由器发现协议（IRDP）。IRDP协议用于某些操作系统发现本地的路由器（例如微软Windows操作系统的大多数版本）。
- 重定向（**Redirection**）——是ICMP的类型5，被路由器用于通知主机去往指定目标的网关，是数据链路上的另一台路由器。假设路由器A和路由器B连接在相同的以太网上，主机X也在以太网上，而且X还把路由器A配置为自己的缺省网关。如果主机向路由器A发送数据包，而路由器A发现该数据包目的地址需通过路由器B才可以到达（即路由器A必须在接收此数据包的端口再次转发此数据包）。路由器A不仅要向路由器B转发数据包，而且还要向主机X发送ICMP重定向消息，通知它如果继续向特定的目标发送数据包，那么请直接将数据包发送给路由器B。示例1-13显示出路由器发送了一个重定向消息。

示例1-13 使用调试功能debug ip icmp，可以看到路由器向主机10.158.43.25发送了一个重定向消息，通知它到达目的地10.158.40.1的正确网关应该是路由器 10.158.43.10

```
Pip#debug ip icmp
ICMP packet debugging is on
ICMP: redirect sent to 10.158.43.25 for dest 10.158.40.1, use gw 10.158.43.10
0
Pip#
```

当数据链路上连接多台路由器时，避免数据包重定向的一个窍门是将每一台主机的缺省网关设置为主机自己的IPv4地址。于是主机对任何目的地址都会发送ARP请求，当目的地址不属于本地数据链路时，合适的路由器将通过代理ARP功能回复请求。使用这种策略避免重定向是有争议的，因为重定向会被减少或消除，但是ARP的流量又增加了。

在IOS软件系统中，缺省状态下重定向功能是打开的。在接口上使用命令**no ip redirects** 可以关闭此功能。

1.6 主机到主机层

TCP/IP协议的主机到主机层的命名恰如其分。尽管网络层负责网络之间的逻辑路径，但主机到主机层是负责两个在完全不同网络[\[18\]](#)上的主机之间的全程逻辑路径。从另一个角度看，主机到主机层向应用提供了一个到协议簇下一层的接口，使应用不必关心它们的数据实际上是如何被传送的。

可以把这种服务比喻为公司的信件收发室。一个包裹被送到收发室，并附有邮寄要求（平信或隔日送到）。提出邮寄要求的人不需要知道或可能不关心实际是怎样邮寄此包裹的。收发室的工作人员将会安排合适的邮寄方式来满足其要求（送邮局邮寄、FedEx、交给骑自行车横穿城镇送快件的人）。

主机到主机层提供两个主要的服务：TCP和UDP。

1.6.1 TCP

传输控制协议（TCP），向应用提供了可靠的、面向连接的服务，详见RFC793。换句话说，TCP提供了一个类似于点到点的连接。

点到点连接有两个特点：

- 仅存在一条到达目的地的路径。进入连接的数据包不会丢失，因为数据包惟一可去的地方就是连接的另一端。
- 数据包到达的顺序与发送顺序相同。

TCP提供了一条看似点到点的连接，虽然实际上这条连接并不存在。TCP利用的Internet层可以提供无连接的、尽力而为转发的服务。这类似于邮政服务。一叠信一旦交给邮递员后，谁也不能保证信件将按照原先叠放的顺序依次送达，也不能保证信件都将在同一天送到，甚至不能保证全部送到。邮政服务仅仅能承诺尽最大努力邮寄这些信件。

同样的，Internet层不保证所有的数据包使用相同的路径，因而也不保证数据包到达时仍旧保持发送时的顺序和间隔或者全部到达。

另一方面，电话呼叫是一个面向连接的服务。数据必须顺序、可靠地到达，否则数据就会作废。像电话呼叫一样，TCP首先必须建立连接，然后是传送数据，当数据传送完成后要拆除连接。

在无连接服务之上，TCP使用了3种基础的机制实现面向连接的服务：

- 使用序列号对数据包进行标记，以便TCP接收服务在向目的应用传递数据之前修正错序的数据包排序。
- TCP使用确认、校验和定时器系统提供可靠性。当接收者按照顺序识别出数据包未能到达或发生错误时，接收者将通知发送者，或者接收者在特定时间内没有发送确认信息，那么发送者就认为在发送结束后数据包没有到达接收方。在这两种情况下，发送者都会考虑重传数据包。
- TCP使用窗口机制调整数据包的流量；窗口机制可以减少因接收方缓冲区满而造成丢失数据包的可能性。

TCP在应用层数据上附加了一个报头，报头包括序列号字段和这些机制的其他一些必要信息，如叫做端口号的地址字段，该字段可以标识数据的源和目标应用程序。为了传送数据，应用数据及附加的TCP报头被封装在IP数据包内。图1-17显示了TCP数据报头字段，示例1-14显示了协议分析器获取的TCP报头信息。

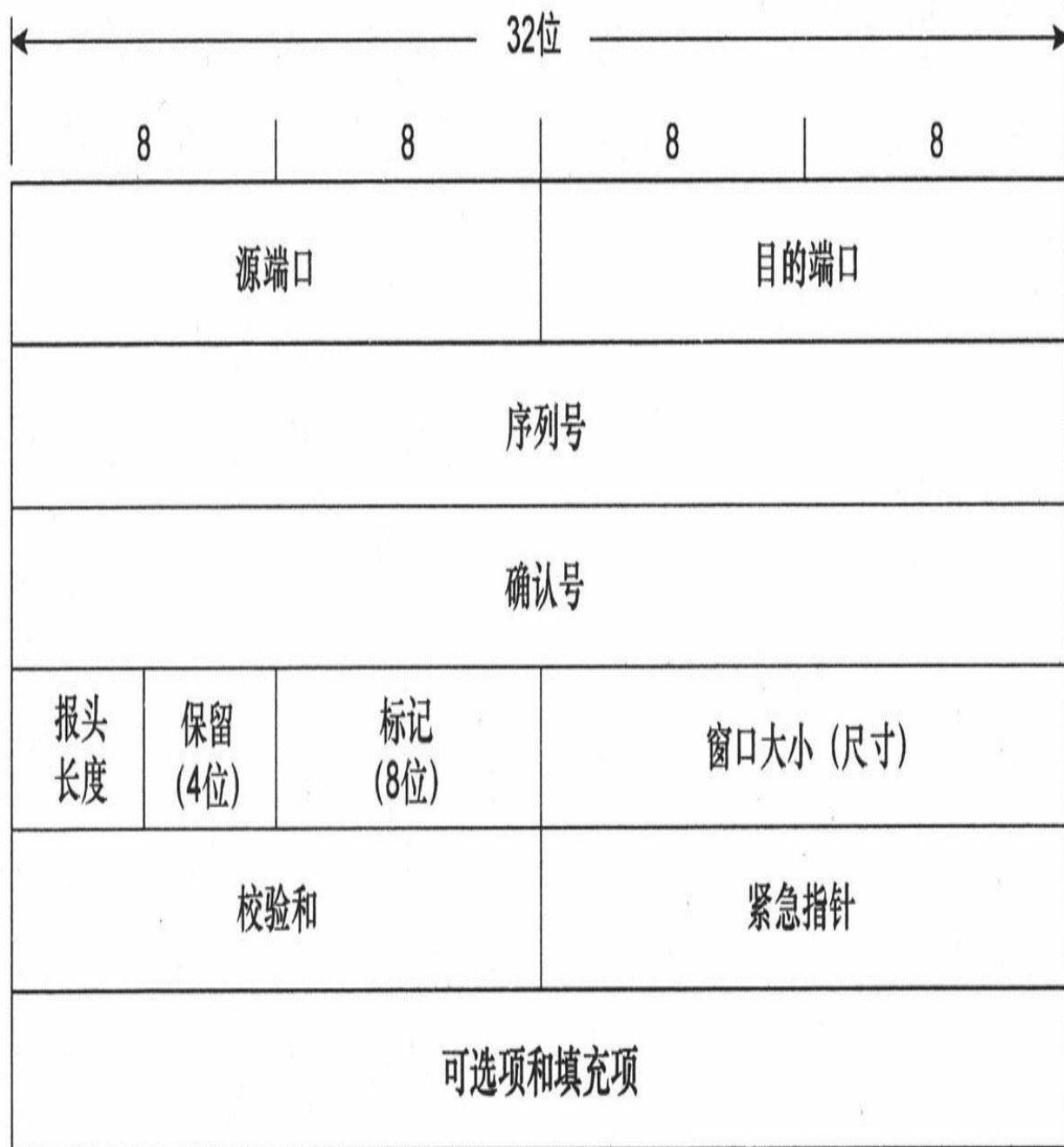


图1-17 TCP报头格式

示例1-14 协议分析器显示出的TCP报头

Ethernet II, Src: 00:0c:41:3c:2b:18, Dst: 00:30:65:2c:09:a6

Internet Protocol, Src Addr: 66.218.71.112 (66.218.71.112),

Dst Addr: 172.16.1.21 (172.16.1.21)

Version: 4

Header length: 20 bytes

Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)

Total Length: 52

Identification: 0xc0b7 (49335)

Flags: 0x04

Fragment offset: 0

Time to live: 50

Protocol: TCP (0x06)

Header checksum: 0x509d (correct)

Source: 66.218.71.112 (66.218.71.112)

Destination: 172.16.1.21 (172.16.1.21)

Transmission Control Protocol, Src Port: http (80),

Dst Port: 60190 (60190), Seq: 288, Ack: 811, Len: 0

Source port: http (80)

Destination port: 60190 (60190)

Sequence number: 288

Acknowledgement number: 811

Header length: 32 bytes

Flags: 0x0010 (ACK)

Window size: 66608

Checksum: 0xb32a (correct)

Options: (12 bytes)

NOP

NOP

```
Time stamp: tsval 587733966, tsecr 1425164062
SEQ/ACK analysis
This is an ACK to the segment in frame: 17
The RTT to ACK the segment was: 0.047504000 seconds
```

- 源端口（**Source Port**）和目的端口（**Destination Port**）——字段长度各为16位，它们为封装的数据指定了源和目的应用程序。像TCP/IP使用其他编号一样，RFC1700描述了所有常用和不常用的端口号。应用程序的端口号加上应用程序所在主机的IP地址统称为套接字（socket）。在网络上套接字惟一地标识了每一个应用程序。
- 序列号（**Sequence Number**）——字段长度为32位，序列号确定了发送方发送的数据流中被封装的数据所在位置。例如，如果本段数据的序列号为1343，且数据段长512个八位组，那么下一数据段的序列号应该为 $1343+512+1=1856$ 。
- 确认号（**Acknowledgment Number**）——字段长度为32位，确认号确定了源点下一次希望从目标接收的序列号。如果主机收到的确认号与它下一次打算发送（或已发送）的序列号不符，那么主机将获悉丢失的数据包。
- 报头长度（**Header Length**）——又叫数据偏移量，长度为4位，报头长度指定了以32位字为单位的报头长度。由于可选项字段的长度可变，所以这一字段标识出数据的起点是很有必要的。
- 保留（**Reserved**）——字段长度为6位，通常设置为0。
- 标记（**Flag**）——包括8个1位的标记，用于流和连接控制。它们从左到右分别是：拥塞窗口减少（Congestion Window Reduced, CWR）、ECN-Echo（ECE）、紧急（URG）、确认（ACK）、弹出（PSH）、复位（RST）、同步（SYN）和结束（FIN）。
- 窗口大小（**Window Size**）——字段长度为16位，主要用于流控制。窗口大小指明了自确认号指定的八位组开始，接收方在必须

停止传输并等待确认之前发送方可以接收的数据段的八位组长度。

- 校验和（**Checksum**）——字段长度为16位，它包括报头和被封装的数据，校验和允许错误检测。
- 紧急指针（**Urgent Pointer**）——字段仅当URG标记置位时才被使用。这个16位数被添加到序列号上用于指明紧急数据的结束。
- 可选项（**Options**）——字段用于指明TCP的发送进程要求的选项。最常用的可选项是最大段长度，最大段长度通知接收者发送者愿意接收的最大段长度。为了保证报头的长度是32个八位组的倍数，所以使用0填充该字段的剩余部分。

1.6.2 UDP

用户数据报协议（UDP）提供了一种无连接、尽力而为传送的数据包转发服务，详见RFC 768。起初，对应用程序宁愿使用不可靠的转发服务，而不用面向连接的TCP服务，感觉很有疑问。然而UDP的优点是不花时间建立连接，直接发送数据。用UDP代替TCP，可以使发送小数据量的应用取得更好的性能优势。

图1-18中给出了UDP的另一个优点：UDP报头长度远远小于TCP报头长度。UDP报头中的源端口和目的端口字段与TCP完全相同；UDP的长度指明了以八位组为单位的整个段长度。校验和包括整个段，但是不同于TCP，在这里，校验和是可选的；当不使用校验和时，此字段全部设置为0。在示例1-15中显示出协议分析器捕捉到的UDP报头。

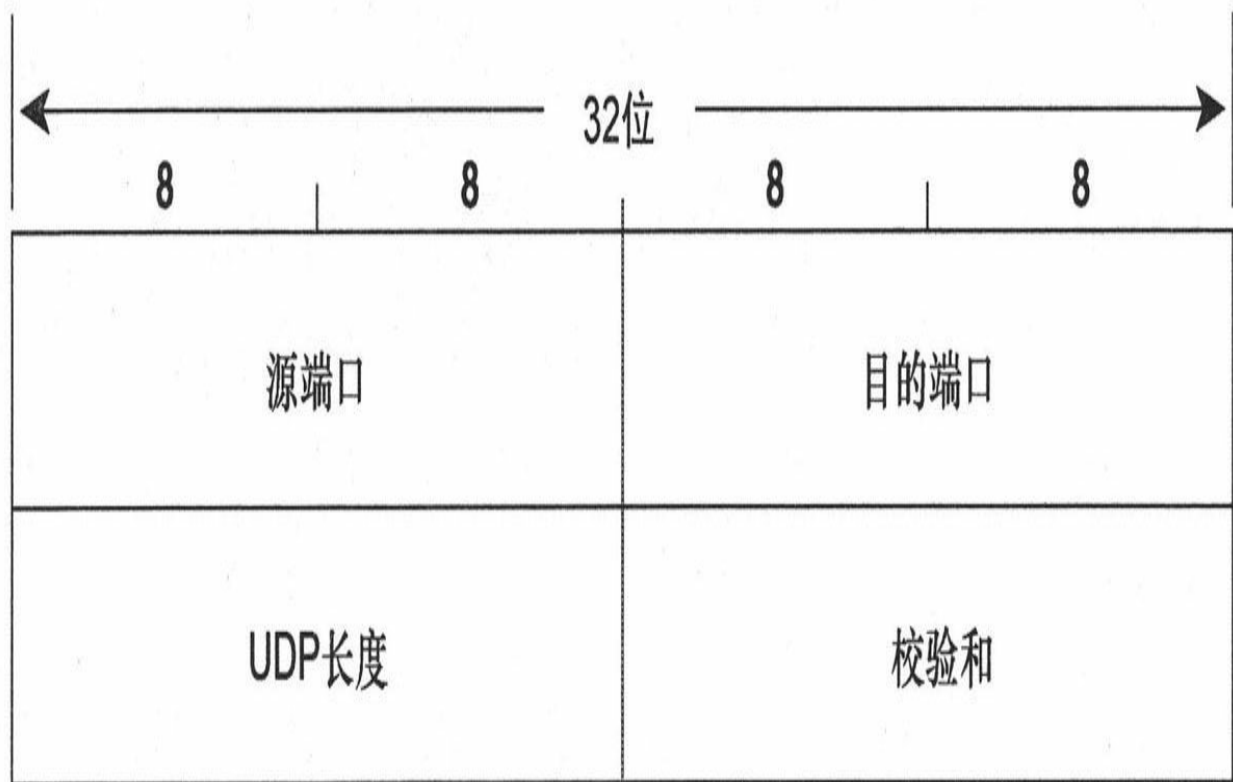
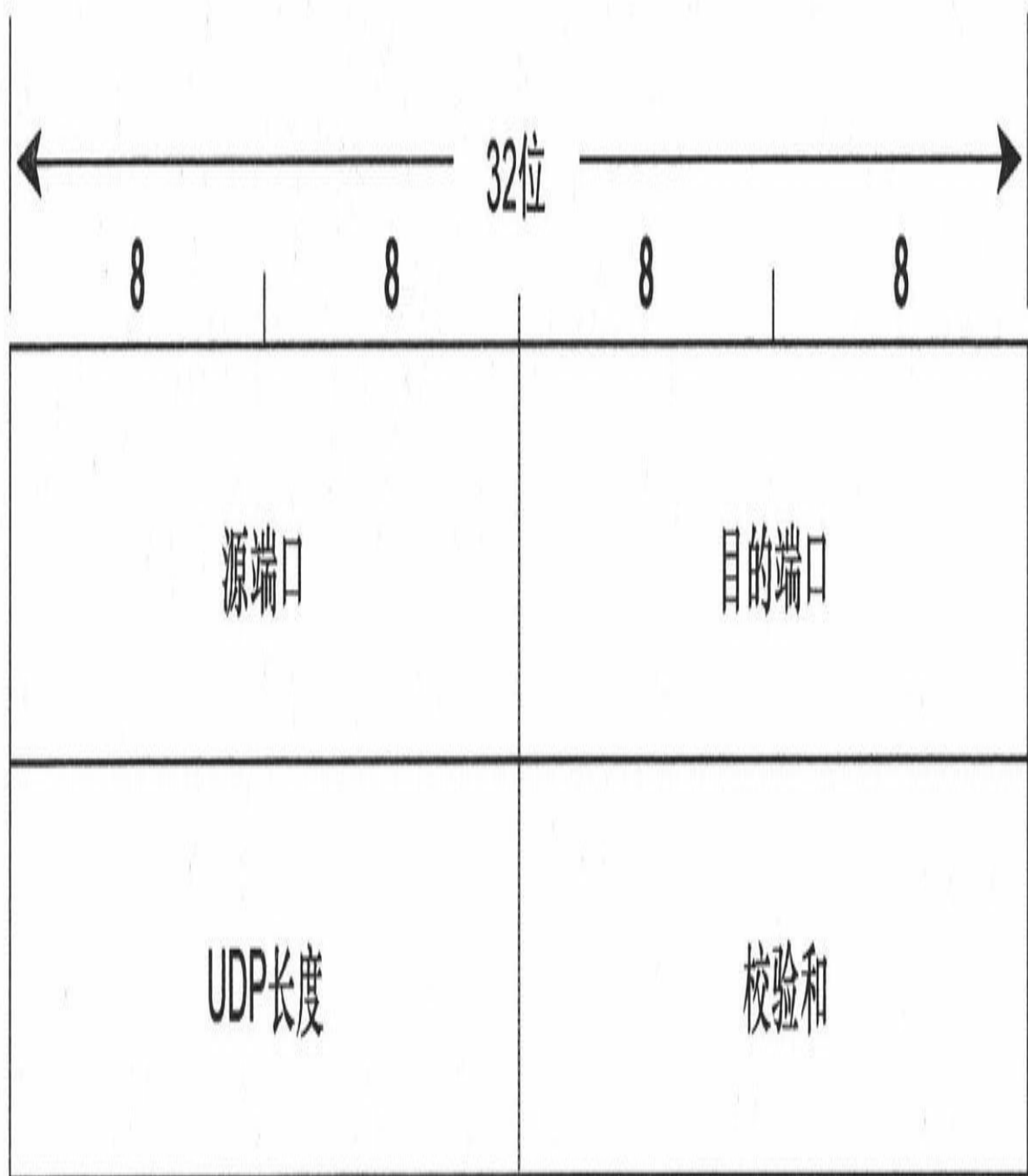


图1-18 UDP报头格式

示例1-15 协议分析器显示的**UDP**报头



1.7 展 望

本章重点讲述了设备的Intenret层（或OSI网络层）的自我标识机制以及Intenret层是怎样映射到网络接口层（OSI数据链路）的。本章还分析了像ARP、ICMP这样的Intenret层协议对路由选择的重要性。下一章将研究IP协议的新版本——IP版本6，它与IPv4的不同之处，以及为什么需要新版本的IP协议的原因。

1.8 总结表：第1章命令总结

命令	描述
arp <i>ip-address hardware-address</i> [alias]	静态地映射IP地址到硬件地址
arp timeout <i>seconds</i>	设置Cisco路由器保留ARP表项的时间值
clear arp-cache	强制从ARP表中删除所有动态表项
debug ip icmp	显示在路由器上出现的ICMP事件
ip address <i>ip-address mask</i> (secondary)	为接口分配IP地址和掩码
ip gratuitous-arp	启动无故ARP特性
ip netmask-format { bit-count decimal hexadecimal }	配置路由器，使路由器可以用位计数、点分十进制和十六进制方式显示IP（地址，掩码）对
ip proxy-arp	启用代理ARP
ip redirects	启用ICMP重定向功能

1.9 推荐读物

Baker, F., ed. “Requirements for IP Version 4 Routers, ”RFC 1812, June 1995.

这篇文章给出了对要运行IP协议的路由器的要求和建议。

Braden, R.,ed. “Requirements for Internet Hosts—Communication Layers,”RFC 1122, October 1989.

RFC 1812的姊妹篇，中心内容是主机。

Comer, D. E. *Internetworking with TCP/IP*, Vol. 1. Englewood Cliffs, New Jersey:Prentice-Hall; 1991.

这本书，就像Perlman的著作一样，是一本经典书。尽管你不一定要把Comer和Stevens的书都读，但是如果都能阅读的话，对你绝不会有坏处。

Stevens, W. R. *TCP/IP Illustrated*, Vol. 1. Reading, Massachusetts: Addison-Wesley; 1994.

这是一本关于TCP/IP的好书。Stevens在深入地介绍协议的同时，针对封二上的网络图提供了大量现实网络的细节。

1.10 复习题

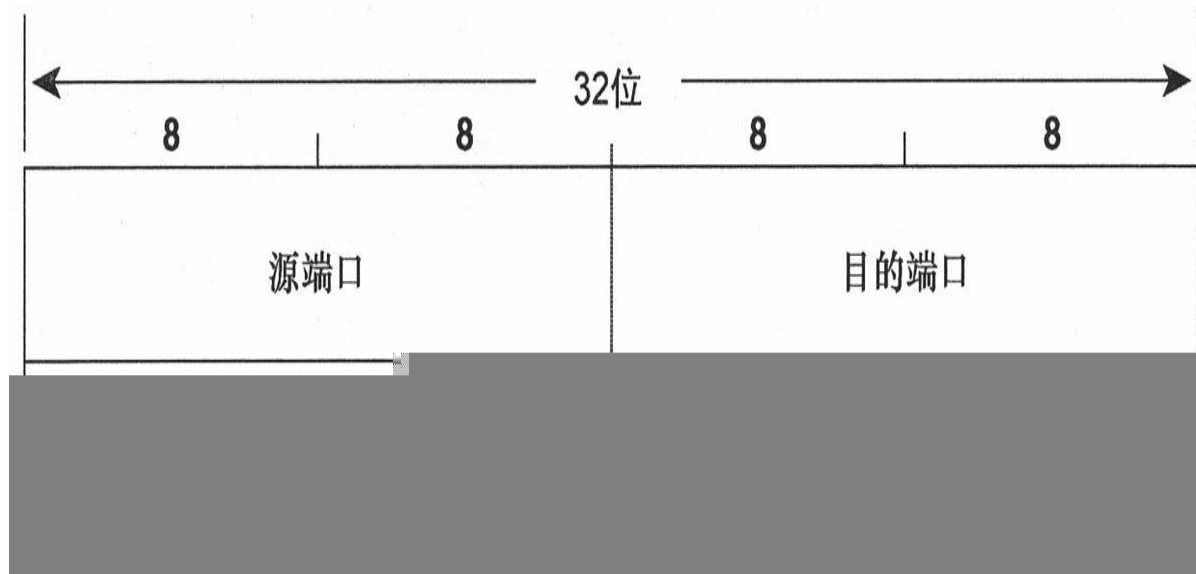
1. TCP/IP协议簇的5个层次是什么？每一层的目的是什么？
2. 目前最常用的IP版本是什么？
3. 什么是分段？IP报头的什么字段用于分段？
4. IP报头中的TTL字段的用途是什么？TTL过程是如何工作的？
5. 什么是首个八位组字节规则？
6. 怎样识别点分十进制表示的A类、B类和C类地址？怎样识别二进制表示的地址？
7. 什么是地址掩码？它是如何工作的？
8. 什么是子网？在IP环境中为什么使用子网？
9. 为什么在有类别路由选择环境中子网位不能全部为0或1？
10. 什么是ARP？
11. 什么是代理ARP？
12. 什么是重定向？
13. TCP和UDP的本质区别是什么？
14. TCP提供面向连接服务的机制是什么？
15. 为了替代ARP，Novell NetWare用设备的MAC地址作为网络地址中的主机部分。为什么IP不能这样做？
16. UDP通过在不连接服务之上提供无连接服务的目的是什么？

1.11 配置练习

1. 首个八位组字节规则指出最高的C类地址是223，而我们知道八位组的最大十进制数是255。因而还有两类地址，一类是D类地址，用于组播，另一类是E类地址，用于实验。其中D类地址的前4位为1110。请问D类地址首个八位组的十进制数的范围是什么？
2. 为10.0.0.0选择一个子网掩码以便至少可以划分出16000个子网，并且每个子网至少拥有700个主机地址。为172.27.0.0选择子网掩码以便至少可以划分出500个子网，并且每个子网至少拥有100个主机地址。
3. 如果C类地址有6个子网位，那么可以划分出多个子网？每个子网有多少个主机地址？这样的子网规划有实际用途吗？
4. 对地址192.168.147.0进行子网划分，子网掩码为28位，请写出所有子网。试给出每个子网的可用主机地址。
5. 对地址192.168.147.0进行子网划分，子网掩码为29位，请写出所有子网。试给出每个子网的可用主机地址。
6. 对地址172.16.0.0进行子网划分，子网掩码为20位，试给出每个子网的可用主机地址（地址按照最低到最高顺序给出）。

1.12 故障诊断练习

1. 根据以下主机地址和子网掩码，试找出每个地址所属的子网，并且找出该子网中的广播地址和可用主机地址的范围：



2. 请问接口上配置IP地址192.168.13.175，掩码255.255.255.240，会有问题吗？如果有，问题是什么？

[1] 除了少数例外，OSI协议簇本身已经成为Internet历史早期的遗留产物。当前OSI协议对于网络技术的贡献看来主要是在对学习网络的学生讲述模块化的协议簇时，可以引用它的参考模型进行说明等有限的用途。当然，IS-IS路由选择协议依然广泛地应用在大型服务提供商和运营商网络中。

[2] BGP是一个应用层的协议，因为它使用TCP 端口传送它的消息；而RIP 协议也是应用层协议的原因是因为使用UDP 接口传递它的消息。其他的路由选择协议如OSPF，称为Internet 层的协议是因为它们直接在IP 数据包中封装它们的消息。

[3] K.Nichols、S.Blake、F.Baker和D.Black,“Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers”, RFC 2474, 1998

年12月。

[4] K.Ramakrishnan, “The Addition of Explicit Congestion Notification (ECN) to IP, ”RFC3168, 2001年9月。

[5] 被分段的数据包不会在数据链路的另一端被重组，而是一直保持分段状态，直至到达最终目的地时才会被重组。

[6] 为了使13位长的分段偏移字段可以表示的最大数据包长度为65535字节，所以使用8个八位组作为本字段的单位。

[7] 正如读者将在第2章中所读到的，在IPv6报头中等价的字段已经重新命名为Hop Limik，以便更加确切地反映它的真正用途。

[8] 点分十进制表示法只用于IPv4地址。在第2章读者将会看到，IPv6地址的表示法完全不同。

[9] 设备使用环回地址（典型的是127.0.0.1）向自己发送流量。发送到该地址的数据将会被直接送回给发送进程，而不会离开此设备。

[10] 注意，223并没有用完第一个八位组中所有可用的数。参见本章最后的配置练习1。

[11] 负责管理和分配IP地址的高级管理机构是亚洲的APNIC、北美的ARIN、中美与南美州的LACNIC,以及EMEA的RIPE。

[12] 事实上，这个地址决不会被分配，因为它属于私有的保留地址；本书中所用到的大多数地址都是保留地址，见RFC1918。保留地址包括：10.0.0.0-10.255.255.255、172.16.0.0-172.31.255.255和192.168.0.0-192.168.255.255。

[13] 1700万个数据链路看上去很多，但是你要考虑到，一个中等规模的企业就可能有許多数据链路。

[14] 所有主机的IP广播地址是所有位全为1：255.255.255.255。特定子网的广播地址是所有主机位全为1；例如，子网172.21.1.0的广播地址是172.21.1.255。最后，对于所有子网的所有主机来说，广播地址是子网为和主机位均为1：172.21.255.255。

[15] 例如，NetWare把设备的MAC地址作为网络层地址的主机部分，这是一种明智之举。

[16] 类似于IP的广播地址，MAC的广播地址也是所有位全部为1：ffff.ffff.ffff。

[17] 在整个TCP/IP协议簇中各字段使用的所有号码都来源于这里：J.Postel和J.Reynolds, “Assigned Numbers,” RFC1700,1994年10月。这本大型文档（230页）是一本很有价值的参考文献，但目前有点过时了。现在有关号码分配的最新列表可以在www.iana.org上查到。

[18] 类似的，主机到主机层也可以看作在传输层之上，功能上等同于OSI的会话层，在两个跨网络的应用之间提供逻辑的、端到端的路径。

本章包括以下主题：

- IPv6地址；
- IPv6包头格式；
- IPv6 扩展报头（Header）；
- Internet消息控制协议第六版（ICMPv6）；
- 邻居发现协议（NDP）。

第2章

IPv6概述

在网络最初开始发展，还未最终发展成为我们现在所称的Internet时，那时的网络还仅仅只限于学术和研究领域。而且，当时在Vint Cerf和Bob Kahn创建有关这些网络的TCP/IP协议簇的时候，更是没有人会预料到Internet能发展为今天这样。在当时看来，所设计的32位地址空间可以提供大约43亿个地址，这看上去好像是不可能耗尽的。

但是，当那些在学院里就使用网络的孩子到了“现实的世界中”时，他们自身就很欣赏基于开放标准上的对等网络创造出的种种可能。从而，越来越多的有价值的网络应用不断地涌现出来；公司连接到某个公共网络的价值也日益得到重视，并开始促成一个商用的Internet网络。在所有这些发生的同时，桌面计算机的使用也变得普及起来，不再是仅仅在办公室使用，更引人注目地是，计算机的使用也在家庭普及开来。但是，那些早期的家庭计算机还没有普遍到把调制解调器作为计算机通用的附属配件，因为那时很少有家庭计算机用户能够意识到个人计算机连接到一个公共网络上的价值和好处。

随着万维网（World Wide Web）的出现，这种情况开始发生改变。突然间，对于非技术领域的用户，轻松地获取和共享信息极大地提高了他们把桌面计算机作为工具的价值。结果，不到20年的时间，Internet已经变成了我们通信、商务活动以及学习的途径和手段。Internet使我们的世界变得更小，同时也对世界政治和经济产生了意义深远的影响。

随着“Internet的大家庭”变得越来越大，它的大小和差异性都呈现出爆炸性的增长，同时也出现了平常令人非常讨厌的东西，例如垃圾邮件、病毒，等等。这样就产生了一个比较严重的技术性问题需要考虑：原来认为可以提供所谓无穷无尽多的IPv4地址现在看来显然已经很有限了。

早在20世纪90年代初期，人们就意识到IPv4地址可能消耗殆尽的问题，当时各方面的专家预测显示，如果IPv4地址的分配按照目前的增长率继续下去，那么在未来短短几年间就将耗尽所有的地址空间。于是，人们就提出了一个新的IP地址版本来解决这个问题，以前在开发阶段这个新

的IP地址版本被称为IP下一代版本（或者称为IPng），而现在一般称为IP协议第六版（或称为IPv6）。但是众所周知，发展一种新的标准需要时间逐步部署，因此，在发展新标准的过程中还需要一种解决IPv4地址耗尽问题的短期方案。

这种短期的解决方案就是网络地址转换（Network Address Translation——NAT），它允许多台主机共享一个或较少的公用IP地址。在NAT设备上，相对外部公共网络的内部网络使用私有IP地址，私有IP地址的使用规则在RFC1918（请求注释）中有详细的描述。读者后面会注意到，在本书的大多数例子中都会使用这种私有IP地址。NAT技术在减缓IPv4地址耗尽问题方面显然非常成功，并在大多数网络设计中已经成为一个标准部分。因而，至今仍然有很多人对发展IP协议新版本的必要性提出质疑。但是，NAT技术的广泛使用把原来具有开放、透明、对等特点的Internet变成了看上去更像一个具有客户-服务器（Client-Server）结构的网络的巨大集合。而用户则只在外围连接到Internet的“边缘层”，Internet向他们提供服务。用户很少对Internet的整体资源作出贡献。更多的从某种经济的角度看，Internet的用户仅仅成为了消费者，而不是生产者。

虽然大多数IPv6协议标准在多年前就已经完成了，但对从IPv4到IPv6协议迁移的巨大兴趣也只是最近才显现出来。这种日益重视使用IPv6协议的背后来自于两个基本的推动力。第一个基本的推动力是对使用诸如移动IP协议（Mobile IP）、服务质量保证、端到端的安全、网格计算（grid computing），以及点到点网络互连等核心概念的新型应用的先见之明。NAT技术遏制了这些领域的创新，因而摒弃NAT技术的惟一手段就是提供充足的并且易于使用的公共IP地址。

促进IPv6协议发展的第二个基本推动力就是拥有众多人口的国家快速的现代化发展，例如中国和印度。一个引人注目的统计数字显示，目前剩余的未分配的IPv4地址数目几乎和中国的人口一样多：大约还有13亿个地址。随着中国国家Internet网络基础设施的急剧扩展，在不久的将来，仅中国就会对现有已经十分紧张的IPv4地址池带来难以忍受的压力。在印度，其人口也接近于中国的人口，不得不继续保留一个具有4~5层NAT技术的网络层次架构，以支持对IP地址的需求。

IPv6协议使用128位的地址替代32位的IPv4地址，这样大约可以产生340万亿亿亿（ 3.4×10^{38} ）个可用的地址。在可预见的未来，这个地址数目将可以满足公共IP地址的需求，也可以解决上面讨论的两个基本推动

力需要的地址需求。[\[1\]](#)

2.1 IPv6地址

IPv6地址与IPv4地址的不同之处不仅仅在于它们的地址长度不同，而在更多的方面都有所不同。快捷地表示它们的“速记”方式也是不同的，它们的表示格式差别非常大，而且它们的功能组织也是不同的。本节我们将为读者介绍这些不同之处。

2.1.1 地址表示法

读者一定已经了解32位的IPv4地址的表示方式了，IPv4地址被分割为4个8位段，其中每个8位段的数字大小在0~255之间，并且每个8位段之间使用英文符号句点“.”来分开，因此有时也使用术语“点分十进制表示法”来专指IPv4地址的这种表示法。

而128位的IPv6地址则被分割成8个16位段来表示，其中每个16位段书写为大小在0x0000~0xFFFF之间的十六进制的数字表示，并且每个16位段之间使用英文符号冒号“:”来分开。例如，下面就是一个IPv6地址的书写方式：

```
3ffe:1944:0100:000a:0000:00bc:2500:0d0b
```

要想记住更多一些像这样表示的地址实际上是几乎不可能的，当然书写这些地址也不是一件令人愉快的事情。幸运地是，有两条规则可以用来简化IPv6地址书写的大小。第一条规则是：

任何一个16位段中起始的0不必写出来；任何一个16位段如果少于4个十六进制的数字，就认为忽略书写的数字是起始的0。

在前面提到的地址例子中，第3、4、5、6和8个分段都包含有起始的0。利用这个地址压缩简化规则，该地址可以书写为：

```
3ffe:1944:100:a:0:bc:2500:d0b
```

这里要注意的是，只有起始的0才可以被忽略掉；末尾的0是不能忽略的，因为这样做会使16位分段变得不确定，你无法确切地判断所省略的0是在所写的数字之前还是在其之后。

另外，还有一个值得注意的地方是，上述的地址例子中的第5个分段全部是0，并且被书写为单个0。事实上，有许多IPv6地址中具有一长串的0。举例如下：

```
ff02:0000:0000:0000:0000:0000:0005
```

这个地址可以简写为以下形式：

```
ff02:0:0:0:0:0:0:5
```

然而，利用第二个规则可以进一步地简化这个地址的书写格式：

任何由全0组成的1个或多个16位段的单个连续的字符串都可以用一个双冒号“::”来表示。

利用这条规则，上面例子中的地址可以表示成如下格式：

```
ff02::5
```

使用这样的方式书写上面这样的地址显然可以增加很多便利。但是在这里要注意的是，这条规则强调的是仅仅对于单个连续不间断的全0字符串分段部分能够用一个双冒号“::”来表示，在一个IPv6地址中使用多于一个以上的双冒号会引起含混不清。下面举一个这样的地址例子作为说明：

```
2001:0d02:0000:0000:0014:0000:0000:0095
```

对于上面这个地址，以下两种地址的缩写方式都被认为是正确的，因为它们都只使用了一次双冒号：

```
2001:d02::14:0:0:95
```

```
2001:d02:0:0:14::95
```

但是，请读者注意，下面这个缩写方式是不正确的，因为它使用了两次双冒号：

```
2001:d02::14::95
```

之所以认为上面这个缩写方式是错误的，是因为它中间的两个全0字符串的长度是含混不清的，从而无法确定它们的长度；它可以表示成下面的任何一种可能的IPv6地址：

```
2001:0d02:0000:0000:0014:0000:0000:0095
```

```
2001:0d02:0000:0000:0000:0014:0000:0095
```

```
2001:0d02:0000:0014:0000:0000:0000:0095
```

不像IPv4协议的前缀（即地址的网络部分）可以通过点分十进制或十六进制地址掩码标识，或可以通过位计数（bitcount）来标识，IPv6协议的前缀始终通过位计数的方式来标识。更确切地说，通过在IPv6地址后面加一个斜线“/”，随后再跟一个十进制的数字来标识一个IPv6地址的起始位有多少位是前缀位。举一个例子，下面这个地址的前缀就是起始的64位：

```
3ffe:1944:100:a::bc:2500:d0b/64
```

当读者需要书写一个IPv6地址的前缀时，也可以使用和IPv4地址一样的书写方式将所有的主机位设置为0。例如：

```
3ffe:1944:100:a::/64
```

一个由全0组成的IPv6地址能够被简单地写成一个双冒号。在本书中，存在两种实例使用了全0的地址。第一个实例就是缺省地址，这将在第12章中讨论，在那里缺省地址表示为全0的形式，并且它的前缀长度也是0：

```
::/0
```

第二个使用全0的IPv6地址的实例是未指定地址（unspecified address）。未指定地址使用在某些邻居发现协议过程中，邻居发现协议将在本章后面的章节中讲述。一个未指定地址就像一个填充器，用来标识一个还未确定的实际IPv6地址。在书写一个未指定地址的时候要注意，它与缺省地址的书写方式是有区别的，它们的前缀长度不同：

```
::/128
```

2.1.2 IPv6的地址类型

IPv6地址存在以下三种类型：

- 单播（Unicast）；
- 任意播（Anycast）；
- 多播（Multicast）。

和IPv4相比，IPv6地址有一个重要的不同，IPv6地址协议中没有广播地址。但是，IPv6地址协议提供了一个包含“全部节点”的多播地址，用来实现与IPv4地址协议中广播地址同样的目的。

1. 全球单播地址

单播地址用来表示单台设备的地址。一个全球单播地址是指这个单播地址是全球惟一的。IPv6单播地址的通用格式如图2-1所示。该格式在RFC3587中有详细地描述，并取代和简化了早期的格式版本，早期的格式将IPv6单播地址分成了顶级聚合（Top Level Aggregator, TLA）、次级聚合（Next-Level Aggregator, NLA）和其他字段。但是，读者应该了解的是，这个被取代的格式其实相对来说是最近才被替代的，因而，读者在一些书籍和资料中可能仍然会碰到对这个旧的IPv6地址格式的描述。

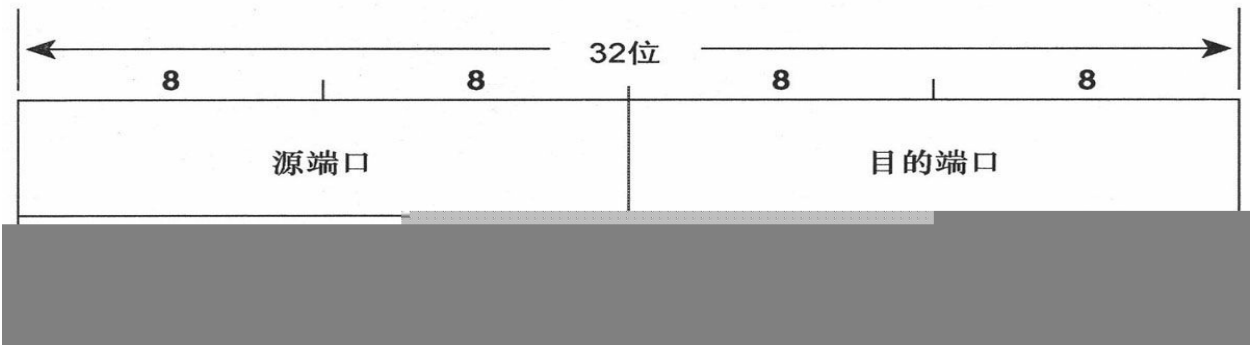


图2-1 IPv6通用的单播地址格式

地址的主机部分被称作接口ID（Interface ID），之所以取这个名字是因为一台主机可以拥有不止一个的IPv6接口，因而使用这样的地址标识主

机的一个接口比标识一台主机本身更加准确。但是，它的精确性也就仅仅到此为止：单个接口也能够拥有多个IPv6地址，并且能够拥有一个附加的IPv4的地址，在这样的实例中，接口ID仅仅表示该接口的几个标识符的其中一个。

除了长度不同外，IPv6地址与IPv4地址协议之间最显著的不同就是子网标识符的位置不同。IPv6地址的子网标识符的位置是地址的网络域的一部分，而不是该地址的主机域的一部分。在IPv4地址分类体系结构的传统概念中，一个地址的子网部分来自于该地址的主机部分，减少了地址的主机位。结果是，IPv4地址的主机部分不仅仅使它的分类产生变化，而且导致用于子网标识的位数产生变化。

使用地址的网络部分作为IPv6子网ID的一个直接的好处就是，所有IPv6地址的接口ID都有大小一致的位数，这就大大地简化了地址的解析复杂度。而且，使用地址的网络部分作为子网ID，会产生一个更加清楚的分工，功能更加清晰：网络部分提供了一台设备到下行专用数据链路的定位，而主机部分提供这条数据链路上该设备的标识。

除了极少数的例外，全球IPv6地址的接口ID都是64位二进制位的长度。同样，除了极少数的例外，子网ID字段都是16位二进制位（如图2-2所示）。一个16位的子网ID字段可以提供65536个不同的子网。使用固定长度大小的子网ID看起来好像有些浪费，因为在大多数实例中远没有使用到这么大容量的子网数。但是，考虑到使用IPv6地址空间的总长度和容易分配、设计、管理以及解析地址的好处，使用固定长度大小的子网ID所带来的浪费也是可接受的。

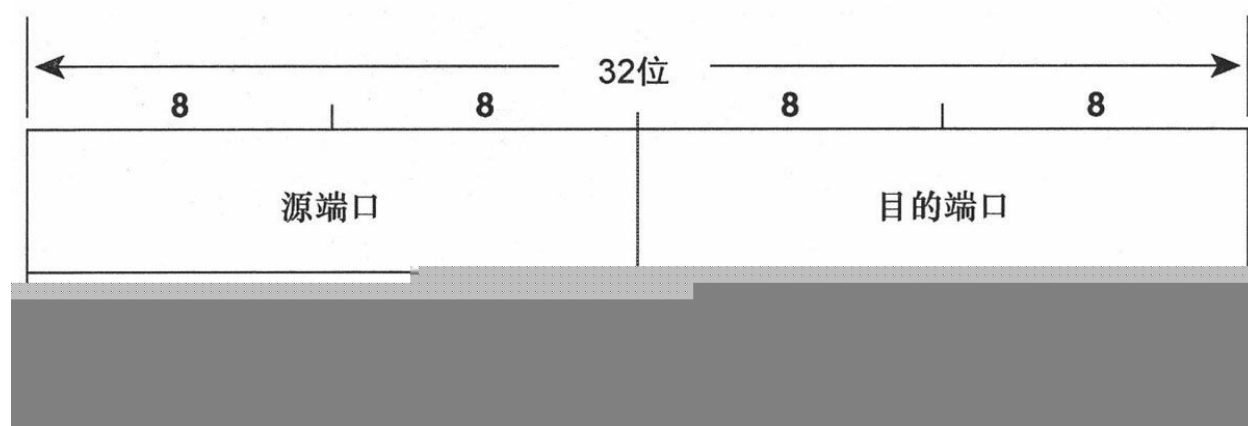


图2-2 全球单播IPv6地址的标准字段大小

Internet地址授权委员会（Internet Assigned Numbers Authority, IANA）和地区Internet注册机构（Regional Internet Registries, RIR）[\[2\]](#)通常把长度为/32或/35的IPv6前缀分配给本地Internet注册机构（Local Internet Registries, LIR）。然后，本地Internet注册机构LIR——通常是大型的Internet服务提供商（ISP），他们再把更长的前缀分配给他们各自的客户。在大多数的实例中，本地Internet注册机构LIR分配的前缀长度都是/48。正如前面所提及的，有一些例外的情况，LIR也可能会分配不同长度的前缀：

- 如果一个客户非常庞大，那么可以分配一个长度小于/48的前缀。
- 如果有一个并且仅有一个子网需要编址，那么可以分配一个长度为/64的前缀。
- 如果有一台并且仅有一台设备需要编址，那么可以分配一个长度为/128的前缀。

2. 标识IPv6的地址类型

IPv6地址起始的一些二进制位指明了该地址的类型。例如，目前所有的全球单播地址的前3位是001。因此，识别全球单播地址的十六进制表示是相当容易的：所有的全球单播地址都是以2或3开头的，这根据全球路由选择前缀的第4位的值而定。举一个例子说明，当前指定分配给6Bone（公共IPv6研究网络，the public IPv6 research network）使用的前缀始于3ffe，而目前由RIR分配的IPv6地址则始于2001。

二进制的数字001预期可以充分满足在未来一段时间内全球单播地址的需要。而其他的一些二进制位的组合则分配用来定义其他的地址类型，其中大多数前导位的组合都是保留的。表2-1列出了目前已经分配的前导位的组合，在后面的章节中将会陆续介绍一些其他主要的IPv6地址类型。

表2-1 IPv6地址类型的高位数字组合

地址类型	高位数字（二进制）	高位数字（十六进制）
------	-----------	------------

未指定	00...0	::/128
::/128	00...1	::1/128
多播地址	11111111	FF00::/8
链路本地单播地址	1111111010	FE80::/10
地区本地单播地址（目前存在争议）	1111111011	FEC0::/10
全球单播地址（当前分配的）	001	2xxx::/4或者 3xxx::/4
保留的类型（未来全球单播地址的分配）	其他所有的	

3. 本地单播地址

当我们谈论全球单播地址的时候，我们就认为这个地址是全球范围使用的。也就是说，这样的地址是全球惟一的，并且能够在全球范围内被路由而无需进行更改。

IPv6也拥有链路本地单播地址（**link-local unicast address**），这种地址是使用范围限定在单条链路上的地址。它的惟一性是仅仅限于所在的链路，并且相同的地址也可能存在于另一条链路上，因此这样的地址离开所在的链路是不可路由的。正如读者在表2-1中所看到的，链路本地单播地址的起始10位永远是1111111010（FE80::/10）。

读者在本章后面的章节中将会看到，链路本地单播地址在像邻居发现协议等功能中是很有用的，邻居发现协议只能在单条链路上进行通信，这在后面的章节中会讲述。链路本地单播地址也允许链路上的设备直接创建IPv6地址和该链路上的其他设备进行通信，而不必给它们分配全球前缀，或者说它们无需知道所在链路的全球前缀。在2.5.3小节中将会讲述在这种情形下如何使用链路本地前缀。

除了链路本地单播地址外，IPv6协议最初还定义了一个地区本地单播地址。地区本地单播地址（**site-local unicast address**）是指仅仅在一个给定的地区区域内该地址是惟一的。在其他地区区域内的设备可以使用相同的地址。因此，一个地区本地单播地址只能在分配该地址的那个地区区域内是可路由的。在IPv6协议中的地区本地地址在功能上和RFC1918中定义的私有IPv4地址有些类似。

地区本地地址的支持者举出这样几个应用。一个非常突出的应用就是，对于那些即使在使用IPv6地址时也希望使用NAT技术的网络人员来说，是为了维持他们自己的地址架构独立于他们的服务提供商。另外，地区本地地址也是几个被提议的IPv6多归路机制的关键。

但是，IETF（Internet工程任务组）的IPv6工作组发现地区本地单播地址也带来了一些困难。一种明显的困难就是这样一个事实：地区区域的定义是含糊的，因为对于不同的网络管理者来说地区的含义是不同的。另一个问题是，当这样的地址被错误地“泄漏到”网络管理者设定的地区范围边界之外时，也像RFC1918定义的IPv4地址一样会涉及到一些管理上的困难。其他一些潜在的问题包括给应用程序和路由器增加了复杂性，因为它们必须识别和复制这些地区本地地址。基于以上这些考虑，并经过一些激烈的争论后，结果是，IPv6工作组在RFC3879中不赞成使用地区本地地址。为了给那些认为地区本地地址有很多优点的人一些信心，又引入了另外一个计划，它具有类似于“比链路本地地址的范围大一些，而比全球地址的范围小一些”的特点，但是到撰写本书时为止还未见到这样一种替代的计划。

如表2-1所示，地区本地单播地址的起始10位是1111111011（FEC0::/10）。

4. 任意播地址

一个任意播地址（Anycast address，也可称为任播地址或泛播地址）表示的更像一种服务，而不是一台设备，并且相同的地址可以驻留在提供相同服务的一台或多台设备中。如图2-3所示，某些服务是由3台服务器提供的，但却是通过IPv6地址3ffe:205:1100::15来进行该服务的所有通告的。

接收到包含该地址通告的路由器不会知道是由3台不同的设备通告给它的。相反，路由器会假定有3条路由到达相同的目的地，并会选择一条代价最低的路由。如图2-3所示，这条路由是到达服务器C的，它的代价是20。

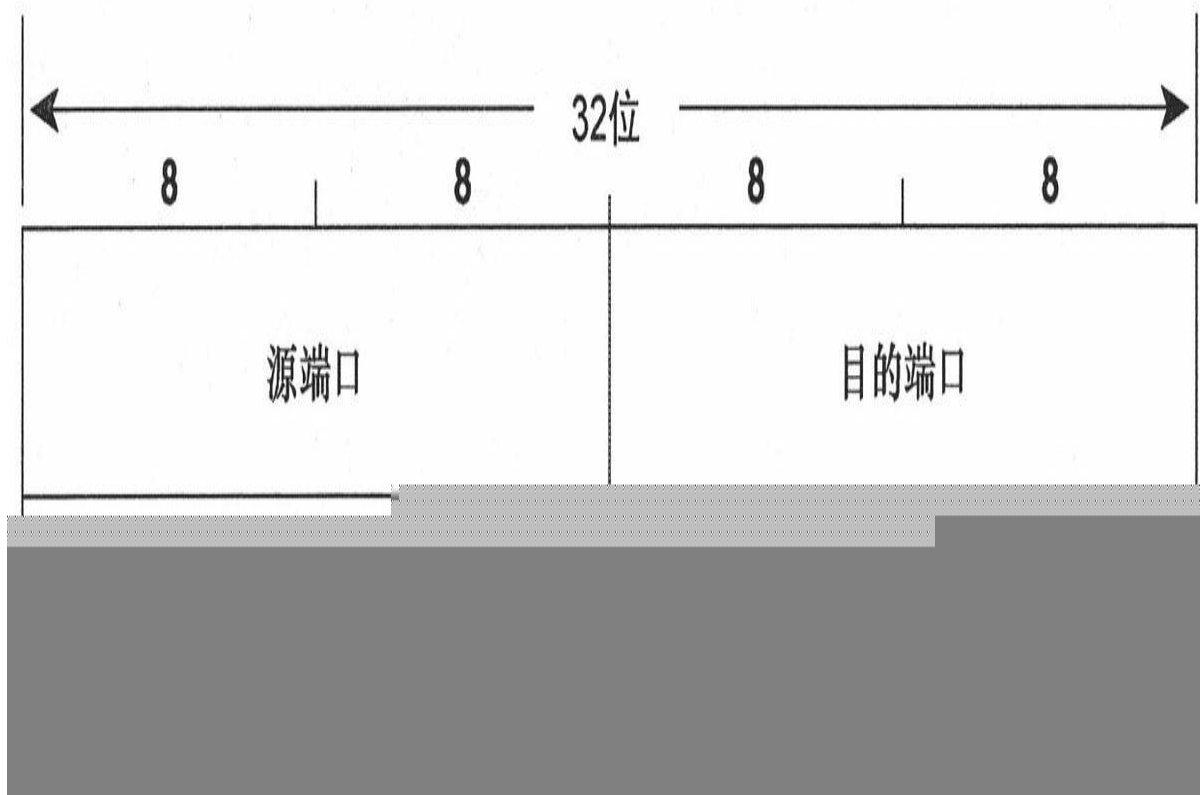


图2-3 一个任意播地址表示的是一种服务，它可能是多台设备

使用任意播地址的好处就是，路由器总是选择到达“最近的”或“代价最低的”服务器的路由。[\[3\]](#)因此，提供一些通用服务的服务器能够通过一个大型的网络进行传播，并且流量可以由本地传送到最近的服务器，这样就可以使网络中的流量模型变得更有效。而且，如果其中一台服务器变得不可用时，路由器能够把路由指向下一台最近的服务器。举例来说，如图2-3所示，如果服务器C因为网络或服务器本身出现故障而变得不可用了，那么路由器就会选择到达服务器A的路径，因为到达服务器A是倒数第二个代价最低的路由。从路由器的角度来看，它选择的是到达同一个目的地最优的路由。

任意播地址仅是根据它们提供的服务功能而定义的，而不是根据它们的格式，而且理论上来说可能是任何范围内的任何一个IPv6单播地址。但是，在RFC 2526中定义了一个保留的任意播地址的格式。任意播地址在IPv4协议的网络中已经使用了一段时间，但是在IPv6协议中它们的定义才被正式化。

5. 多播地址

多播地址标识的不是一台设备，而是一组设备——一个多播组（multicast group，或称为多播群）。发送给一个多播组的数据包可以由单台设备发起。因此，一个多播数据包通常包括一个单播地址作为它的源地址，一个多播地址作为它的目的地址。在一个数据包中，多播地址从来不会作为源地址出现。

一个多播组的成员可能只有一台单个的设备，也可能甚至是该网络上所有的设备。事实上，IPv6协议并不像IPv4协议那样有一个保留的广播地址，而是有一个保留的包含所有节点的多播组，实际上做相同的事情：所有接收它的设备都是属于该多播组。

多点传送实际上是IPv6协议的一个基本的操作，特别是对于即插即用特性的一些功能，例如路由器发现和地址自动配置等，这些功能是邻居发现协议的一部分，邻居发现协议将在本章后面讲述。

IPv6多播地址的格式如图2-4所示。多播地址起始的8位总是全1，并且后跟的4位被指定作为标记位。这些标记位的前3位目前没有使用，全部设置为0。第4位用来指出这个地址是一个永久的、公认的地址（设为0），还是一个管理分配使用的暂时性的地址（设为1）。接下来的4位数字表示该地址的范围，如表2-2所示。表2-3中显示了几个保留的、公认的IPv6多播地址，所有这些地址都属于链路本地的范围。由于多播组总是一组独立的节点，因而在多播地址中的子网字段是不需要的，或者说是没有意义的。而最后的112位用来作为组ID（Group ID），标识各个不同的多播组。目前的用法是设置前面的80位为0，而只使用后面的32位。

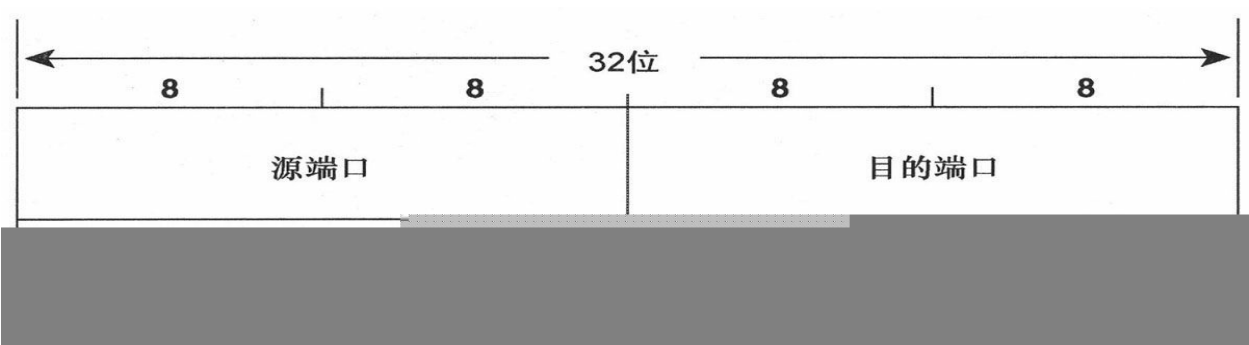


图2-4 IPv6多播地址的格式

表2-2 多播地址的范围

范围字段的值	范围
0x0	保留
0x1	节点本地（Node-Local）
0x2	链路本地
0x5	地区本地
0x8	组织本地（Organizaion Local）
0xE	全球的
0xF	保留

表2-3 公认的**IPv6**多播地址举例

地址	多播组
FF02::1	所有的节点
FF02::2	所有的路由器
FF02::5	OSPFv3路由器
FF02::6	OSPFv3指定路由器
FF02::9	RIPng路由器
FF02::A	EIGRP路由器
FF02::B	移动代理（Mobile Agents）
FF02::C	DHCP服务器/中继代理
FF02::D	所有的PIM路由器

6. 嵌入的**IPv4**地址

有几种转换技术——将**IPv4**地址的网络转换成**IPv6**地址的技术，或者另外一种让两者共存的技术——要求**IPv4**地址在**IPv6**地址环境中进行通信。这些不同的技术详细说明了**IPv4**地址如何嵌入到**IPv6**地址中的细节，以及如何实现在128位的**IPv6**地址中寻找32位的**IPv4**地址的技术。但是，读者也会发现，多数技术使用惟一的格式来表示它们的地址，以便允许读者识别嵌入的**IPv4**地址。例如，一个嵌入了**IPv4**地址10.23.1.5的**IPv6**地址是：

FE80::5EFE:10.23.1.5（一个ISATAP地址）

::FFFF:10.23.1.5和::FFFF:0:10.23.1.5（SIIT地址）

FEC0:0:0:1::10.23.1.5（TRT地址）

在上述的每个例子中，IPv4地址都是位于IPv6地址的最后32位，并使用点分十进制表示法表示。

其他使用嵌入IPv4地址的转换技术都不是使用点分十进制表示，而是把IPv4地址转换成十六进制编码表示。例如，6to4技术就是使用十六进制表示。地址10.23.1.5在十六进制中的表示是0A17:0105，所以嵌入了地址10.23.1.5的一个6to4前缀表示为：

2002:0A17:0105::/48

本卷书的内容并不涵盖这些转换技术，所以读者在本书中不太可能会看到这些地址表示方式。在这里之所以提到它们，仅仅是因为读者如果从事IPv6协议方面的工作时很可能会遇到像这样的地址。

2.2 IPv6包头格式

IPv6包头（Packet Header）的格式如图2-5所示。这和IPv4的数据包头部有些明显的相像，也有些或明显的或细微的不同之处，IPv4的数据包头部请参见前面章节中的图1-2所示。

- 版本（**Version**）——和IPv4的报头（Header）一样，是一个4位的字段，用来指出IP协议的版本。当然，在这里它被设置为0110，表明是版本6。
- 流量类别（**Traffic Class**）——是一个8位的字段，这相当于IPv4协议中的ToS字段。但是，考虑到ToS字段这些年的发展，现在都用来做区分服务等级（Differentiated Class of Service, DiffServ）了。所以，即使这个字段和旧的ToS字段有些相似，它们的名字要比所传送的值更能确切地反映目前的用处。

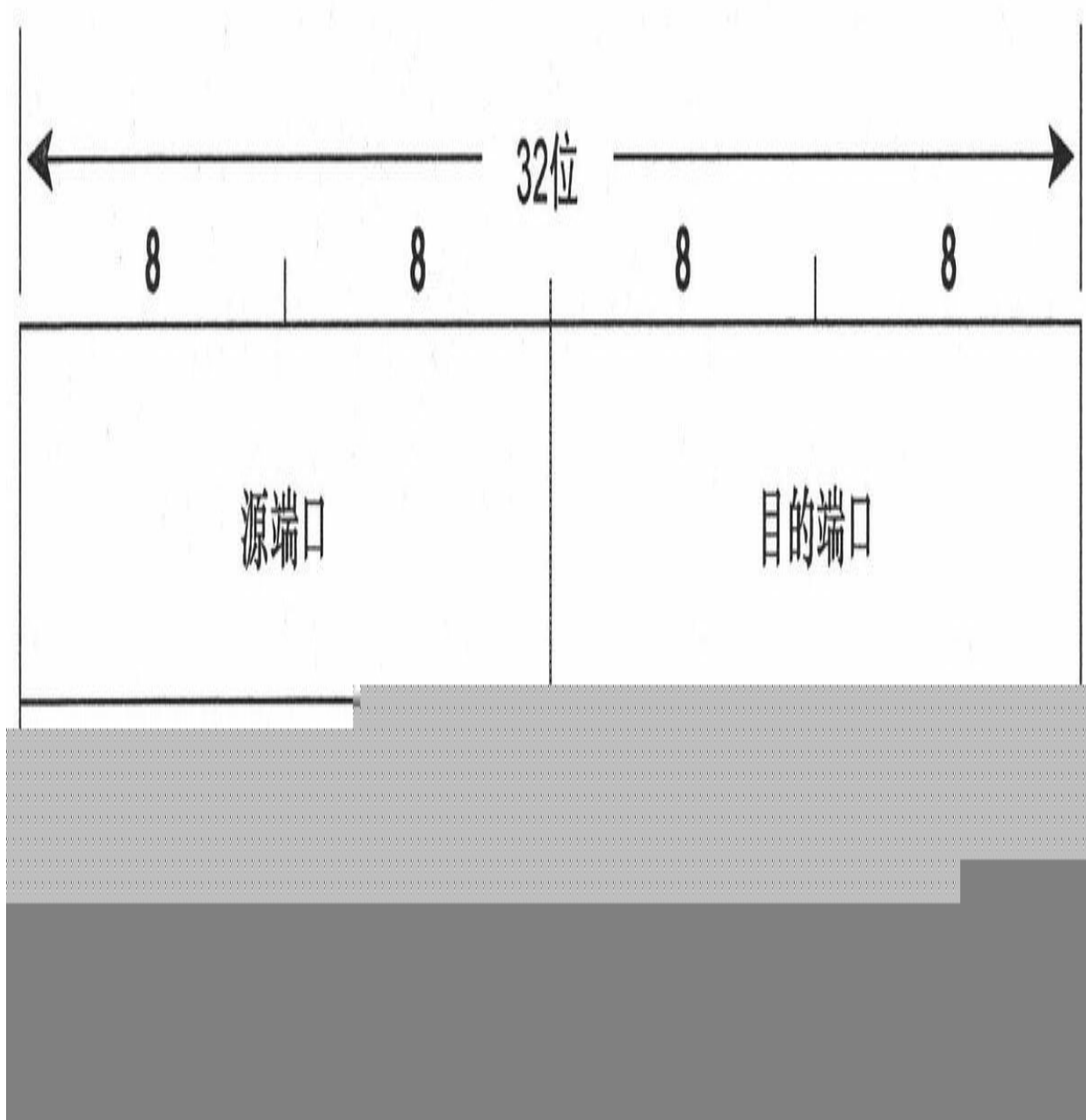


图2-5 IPv6的报头

- 流标签 (**Flow Label**) ——是IPv6协议独有的字段，长度为20位。这个字段的设置目的是允许为特定的业务流打上标签；也就是说，数据包不仅仅始发于相同的源和到达相同的目的地，而且在源和目的地都属于相同的应用。区分不同的流可以带来几方面的优点，从可以提供更精细的服务类别区分的颗粒，到在平衡业务流量通过多条路径时可以确保属于同一个流的数据包能够总是转发到相

同的路径上去，以便避免对数据包进行重新排序。流（或更精确地称为微分流，microflows）可以用源地址、目的地址加上源和目的端口的组合来确定。

但是，为了识别源和目的端口，路由器必须能够看到IP报头的上层——TCP或UDP（或其他传输层协议）报头——这就增加了转发处理的复杂性，并可能会影响路由器的性能。由于IPv6的扩展报头，在IPv6的数据包里搜索传输层报头尤其会产生问题，扩展报头将在下一小节中讲到。一台IPv6的路由器必须分步通过可能存在的多个扩展报头来搜索传输层报头。

在发起一个数据包时，加上合适的流标签字段，路由器就能够识别一个流，而不必进一步地查找数据包头部。但是，在撰写本书时，有关如何使用流标签字段的完整说明仍然还在争论，因此目前路由器忽略这个字段。尽管如此，它还是作出了这样一个许诺——允许IPv6可以为像IP语音（VoIP）这样的应用提供比IPv4更好的服务质量保证（QoS）。

- 有效载荷长度（**Payload Length**）——用来指定数据包所封装的有效载荷的长度，以字节计数。请读者回忆一下第1章，因为IPv4的报头包含可选字段和填充字段，因而它的报头长度是可变的。因此，为了得到IPv4数据包的有效载荷长度，必须用总长度字段的值减去报头长度字段的值。而另一方面，IPv6数据包头部长度总是固定的40字节，而且单从有效载荷长度字段就足以能得到有效载荷的起始和结尾了。

在这里也请读者注意，尽管IPv4的总长度字段是16位的，但是IPv6的有效载荷长度字段却是20位。这意味着该字段能够指定更长的有效载荷（1048575字节，相对IPv4中只有65535字节）。因此，IPv6数据包本身在理论上来说具有传送更大的有效载荷的能力。

- 下一报头（**Next Header**）——指出了跟随该IPv6数据包头部后面的报头。在这里，这个字段和IPv4协议报头中的协议字段非常类似。事实上，当下一报头字段是一个上层协议报头时，其将用于同样的目的。与IPv4中的字段一样，这个字段也是8位的。但是，在IPv6协议中，跟随在该数据包头部后面的报头可能不是一个上层协议报头，而是一个扩展的头部（在下一节中讲述）。因此，从下一头部字段的命名上就可以反映出它具有更广泛的功能范围。

- 跳数限制（**Hop Limit**）——这个字段和IPv4协议中生存时间

（TTL）字段在长度（都是8位）和功能上都是非常一致的。正如第1章中所讲到的，TTL字段的最初设想是，在数据包转发过程期间，当在路由器上排队等待时就用该字段的值减去相应的以等待的秒数为单位的计数值，但这一功能从来没有实现过。相反，路由器直接对TTL值进行减1，而不管该数据包在这台路由器上排队等待了多长时间（况且，在现代网络中，数据包在任何一个地方排队等待接近1s的时间都是非常不正常的）。因此，TTL事实上总是被用来作为衡量一个数据包到达目的地的路径中所能跨越的最大路由器跳数的工具。如果TTL值减少为0，那么该数据包就被路由器丢弃。跳数限制的功能也完全相同，但它的命名更加贴近了它的功能特点。

- 源地址和目的地址（**Source and Destination Address**）——这和IPv4协议中的源地址和目的字段是一样的，当然，在IPv6协议中，这些字段是128位的长度而已。

在IPv6报头中，很明显缺少的一个字段是IPv4报头中包含的校验和字段。在现代传输介质的可靠性全面提高的今天——当然无线传输或许是例外——由于上层协议通常携带它们自己的错误校验和恢复机制，IPv6报头本身的校验和就体现不出太多的价值了，因而就被去掉了。

2.3 IPv6扩展报头

对比图2-5中的IPv6报头和图1-2中的IPv4报头，读者可以看出，虽然源地址和目的地址字段的长度都是报头的4倍，但是IPv6报头本身的长度并不比IPv4报头长：IPv6报头长度为40字节，而IPv4报头最小长度为20字节。如果IPv4的可选项字段也用来进行扩展应用（虽然这不常见），那么IPv4报头的长度实际上比IPv6报头大。

读者会注意到，除了可选项字段，其他的字段并不是很常用到，例如那些与分段有关的字段，从而在IPv6报头里就去掉了那些字段。因此，给定了固定长度并且排除了所有不携带每个数据包转发时所必要信息的字段，IPv6报头变得更加简洁和有效。

但是，如果我们需要使用这些IP特性的某个可选项，例如分段、源路由选择或认证，我们又该怎么做呢？于是就在IPv6协议中，提供了一项可选的功能——扩展报头（extension header），在需要提供这些功能时可以添加在报头之后。例如，如果需要使用源路由选择、分段和认证等可选功能，那么就可以把它们各自需要增加的功能信息加载到3个扩展报头当中，就像图2-6所显示的那样。因为这些报头，IPv6数据包可以在以下两个方面提高效率：

- 数据包仅仅需要传送各自数据包所需要的信息，不需要传送用不到的字段。
- 可以通过定义新的扩展报头添加到IPv6数据包中来增加新的可选功能。

每一个扩展报头都像IPv6报头一样，有一个下一报头字段。因此，每一个报头都会告知是哪一个报头跟在它的后面。表2-4中显示了当前定义的扩展报头和它们下一报头的值。例如，如图2-7所示，IPv6报头中下一报头字段的值表明它的下一个报头是一个路由选择扩展报头（43），这个报头的下一报头字段表示它的下一个报头是一个分段扩展报头（44），依此类推。最后一个扩展报头AH表示它的下一个报头是一个TCP报头（协议号为6）。

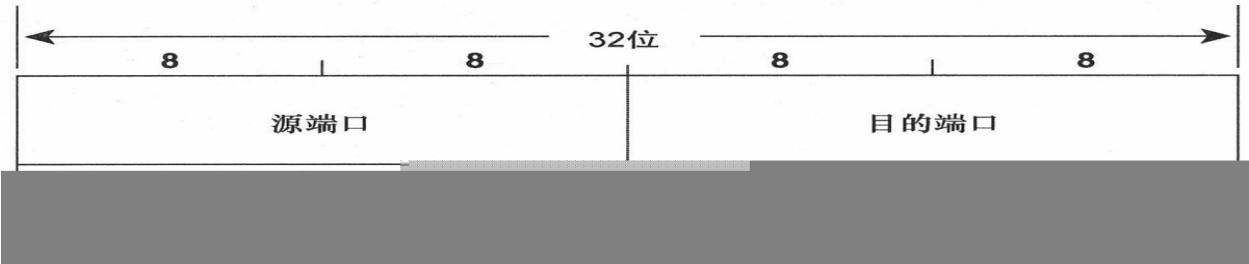


图2-6 扩展报头允许IPv6数据包传送该数据包需要的所有信息，但传送的是该数据包仅仅需要传送的信息

表2-4 下一报头的值

报 头	下一报头的值
逐跳可选项	0
路由选择	43
分段	44
封装安全有效载荷 (ESP)	50
认证报头 (AH)	51
目的地可选项	60
TCP/IP协议	由协议定义的协议号的值（例如TCP=6, UDP=17, OSPF=89,等等）
没有下一报头	59

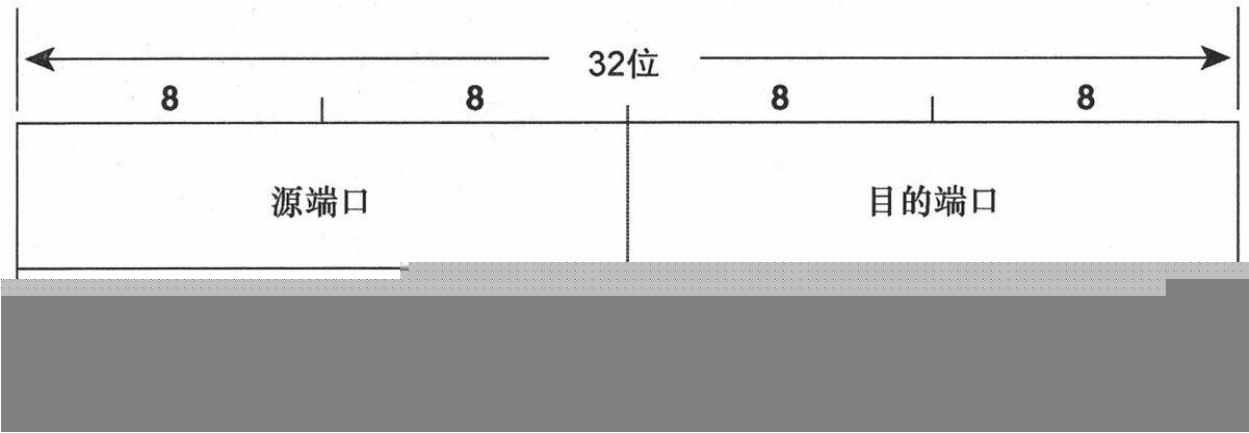


图2-7 IPv6报头中的下一报头字段和每一个扩展报头指定的跟在其后的报头

在RFC 1883中描述了每一个扩展报头的格式。但是概括来说，每个扩展报头的功能如下：

- 逐跳可选项（**Hop-By-Hop Options**）——传送必须被转发路径中的每一个节点都检验处理的信息。例如，路由器告警和超大包有效载荷选项等。
- 路由选择（**Routing**）——通过列出在到达目的地的路径中数据包所要经过的节点列表来提供源路由选择的功能。
- 分段（**Fragment**）——是指在一个数据包被分段时用来为接收节点重组数据包提供必要的信息。在IPv4和IPv6数据包中有一个重要的不同是，只有发起该数据包的节点能够对数据包进行分段；而IPv6路由器对数据包并不分段。因此，发起该数据包的节点要么必须使用路径MTU发现（Path MTU Discovery, PMD）来得到该数据包到达目的地的路径上最小的MTU值，要么就从不发出大于1280字节的数据包。PMD将在下一小节中讲述。IPv6协议规定运行IPv6的所有链路都必须能够支持最小1280字节大小的数据包。因此，发起数据包的节点如果可以选择的话，可以利用最小长度大小选项，而不用PMD。
- 封装安全有效载荷（**Encapsulating Security Payload, ESP**）——用于有效载荷的加密封装。
- 认证报头（**Authentication Header, AH**）——用于数据包必须在源与目的节点之间进行认证的情况。
- 目的地可选项（**Destination Options**）——用于传送仅仅被目的节点，或者可能是路由选择报头中列出的节点检验处理的信息。

如果使用扩展报头的话，在RFC 1883中也规定了它们应该出现的顺序。这里严格规定，如果使用逐跳可选项的话，它必须直接跟在IPv6报头的后面，以便于它能够被必须处理它的传输节点很容易的发现。建议的扩展报头顺序如下：

1. IPv6报头。
2. 逐跳可选项。

3. 目的地可选项（只有在路由选择报头中指定的中间路由器才必须处理这个报头）。
4. 路由选择。
5. 分段。
6. 认证。
7. 封装安全有效载荷。
8. 目的地可选项（只有最后的节点必须处理这个报头）。
9. 上层报头。

2.4 ICMPv6

正像IPv4一样，IPv6也需要一个控制协议来交换与处理错误和信息的信息。并且和IPv4一样，它也使用ICMP来实现这一功能。但是在IPv6中使用ICMP和在IPv4中使用ICMP并不一样。在IPv4协议中ICMP使用的协议号是1，而在IPv6协议中ICMPv6使用的下一报头的值为58。

ICMPv6（Internet消息控制协议第六版）是在RFC 2463中规定的。在RFC中定义的很多功能都和IPv4中ICMP的定义相同；但是，在ICMPv6中也有很多ICMP消息意义并不相同，例如源抑制（Source Quench）和时间戳（Timestamp）消息。

比较图2-8中显示的ICMPv6报头和图1-28显示的ICMP报头，读者能够看到它们基本上是相同的。并且ICMPv6像ICMP一样使用一组类型与代码值的组合来标识一般的类型和它们的子类型。在RFC 1885中给出了这些值的定义，如表2-5中所列。

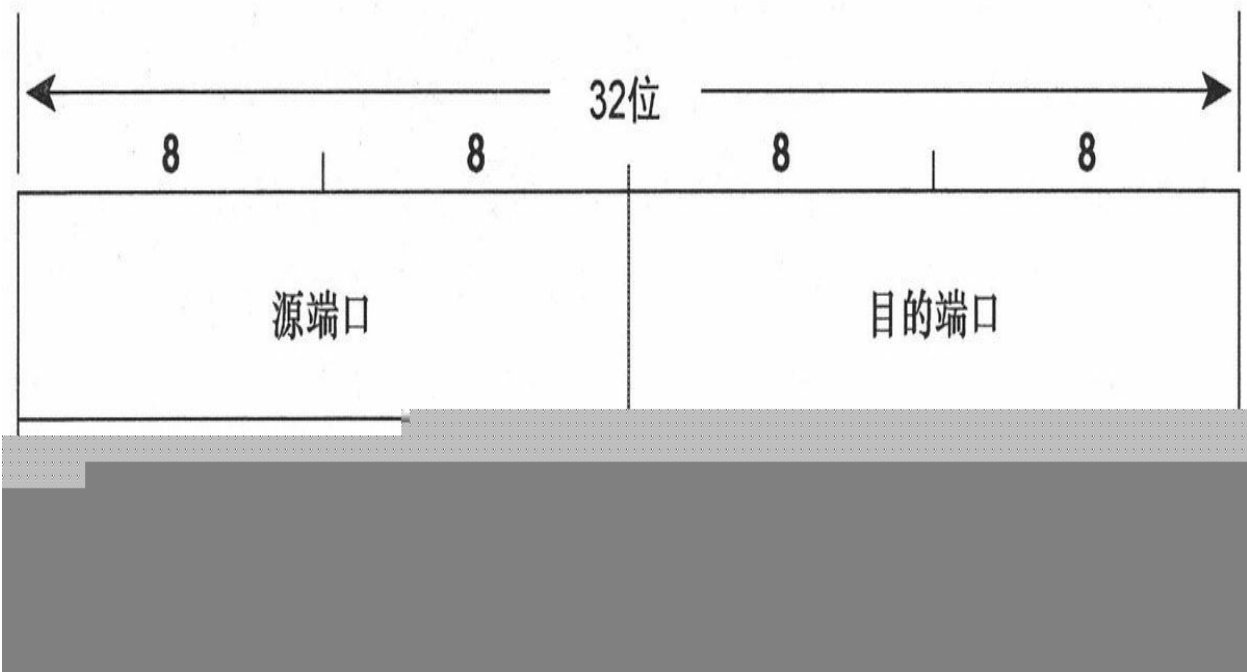


图2-8 ICMPv6报头的格式

表2-5 ICMPv6消息的类型域和代码域

类型	代码	消息
1		目的地不可达
	0	没有到达目的地的路由
	1	和目的地的通信被管理员禁止
	2	不是邻居
	3	地址不可达
2	4	端口不可达
	0	数据包长度太大
3		超时
	0	在传输时超出跳数限制
	1	分片的重组时间超时
4		参数问题
	0	遇到错误的报头字段
	1	遇到不可识别的下一报头类型
	2	遇到不可识别的IPv6选项
128	0	ECHO请求
129	0	ECHO答复
130	0	组成员请求
131	0	组成员报告
132	0	组成员减少

ICMPv6协议除了基本的错误和信息控制功能外，还包括一些使用ICMPv6消息的机制。例如，在前面章节里提到的路径MTU发现机制会发送长度越来越大的数据包到目的节点。当一个给定的数据包长度超过到达目的节点的路径上最小的MTU时，该数据包将被丢弃，并发送一个数据包长度太大的消息给源地址，从而源地址节点就可以知道这条路径上最小的MTU大小。和IPv4一样，Echo请求和Echo答复消息用于Ping的功能当中。

然而，ICMPv6除了基本的错误和信息消息外，还使用了一组单独的由基本的IPv6协议ICMPv6消息：邻居发现协议，这将在下面讲述。

2.5 邻居发现协议（NDP）

IPv6协议除了它显著地增加了地址空间外，一个最显著的特征就是它的即插即用特性。邻居发现协议（Neighbor Discovery Protocol, NDP）就是使用以下的功能来实现这些即插即用特性的协议：

- 路由器发现（**Router Discovery**）——当一个节点连接到一个IPv6的链路上时，它能够发现本地的路由器，而不必借助动态主机配置协议（DHCP）。
- 前缀发现（**Prefix Discovery**）——当一个节点连接到一个IPv6的链路上时，它能够发现分配给该链路的前缀。
- 参数发现（**Parameter Discovery**）——节点能够发现它所相连的链路的参数，例如链路的MTU和跳数限制等。
- 地址自动配置（**Address Autoconfiguration**）——节点能够确定它的完整地址，同样也不需要借助DHCP协议。
- 地址解析（**Address Resolution**）——节点不需要利用地址解析协议（ARP）就能够发现所连接链路上其他节点的链路层地址。
- 下一跳确定（**Next-Hop Determination**）——一条链路上的节点能够确定到达目的节点的下一跳链路层节点，或者是本地的目的节点，或者是到达目的节点的路由器。
- 邻居不可达检测（**Neighbor Unreachability Detection**）——节点能够检测到链路上的邻居何时不再可达，在这里邻居可能是其他主机也可能是一台路由器。
- 地址冲突检测（**Duplicate Address Detection**）——节点能够检测到它所要使用的地址是否已经被所在链路上的其他节点占用。
- 重定向（**Redirect**）——对于非连接（off-link）的目的节点，路由器能够通过重定向消息通知主机存在比它自己更好的下一跳路由。该重定向功能在IPv4协议中是ICMP基本功能的一部分，但是在IPv6协议里被重新定义为邻居发现协议的一部分。

NDP消息通常应该在链路本地的范围内收发，因此，封装NDP消息的数据包也始终使用IPv6链路本地地址，或者链路本地范围内的多播地址。为了增加更进一层的安全性，承载所有NDP消息的IPv6数据包的跳数限制为255。如果所收到的数据包中有跳数限制的值小于255的，那么就说明这个数据包最少已经经过了一台路由器，因而该数据包将被丢弃。这种做法可以阻止NDP受到来自不与本地链路相连的源节点的攻击或欺骗。

2.5.1 NDP消息

NDP是在RFC 2461中定义的，为了完成某些功能，它使用ICMPv6协议来交换一些必要的消息，具体来说，在RFC2461中详细说明了以下5个新的ICMPv6消息：

- 路由器通告（**Router Advertisement, RA**）消息 —— 路由器通告消息由路由器发起，用来通告这些路由器的存在和链路细节的参数，例如，链路前缀、链路MTU，以及跳数限制等。这些消息周期性地发送，也用于答复路由器请求消息。
- 路由器请求（**Router Solicitation, RS**）消息 —— 路由器请求消息由主机发起，用来请求路由器发送一个RA。
- 邻居请求（**Neighbor Solicitation, NS**）消息 —— 邻居请求消息由节点主机发起，用来请求另一台主机的链路层地址，也用来实现诸如地址冲突检测和邻居不可达检测的功能。
- 邻居通告（**Neighbor Advertisement, NA**）消息 —— 节点发送一个邻居通告消息来响应邻居请求消息。如果一个节点改变了它的链路层地址，那么它能够通过发送一个未请求的邻居通告消息来通告这个新地址。
- 重定向（**Redirect**）消息 —— 重定向消息的使用和IPv4协议中ICMP的用法是相同的，它们只不过从基本的ICMPv6协议中抽取了一部分移到了NDP当中。

如图2-9所示，图中显示了路由器通告消息的格式。它的ICMPv6类型是134，代码是0。封装这个RA消息的IPv6数据包的源地址总是始发这个数据包的接口的IPv6链路本地地址。假如这个RA消息周期性地发送，

它的目的地址就是所有节点的多播地址（FF02::1），或者 如果这个RA消息是为了响应一台路由器请求消息，则它的目的地址是发起请求的节点的链路本地地址。

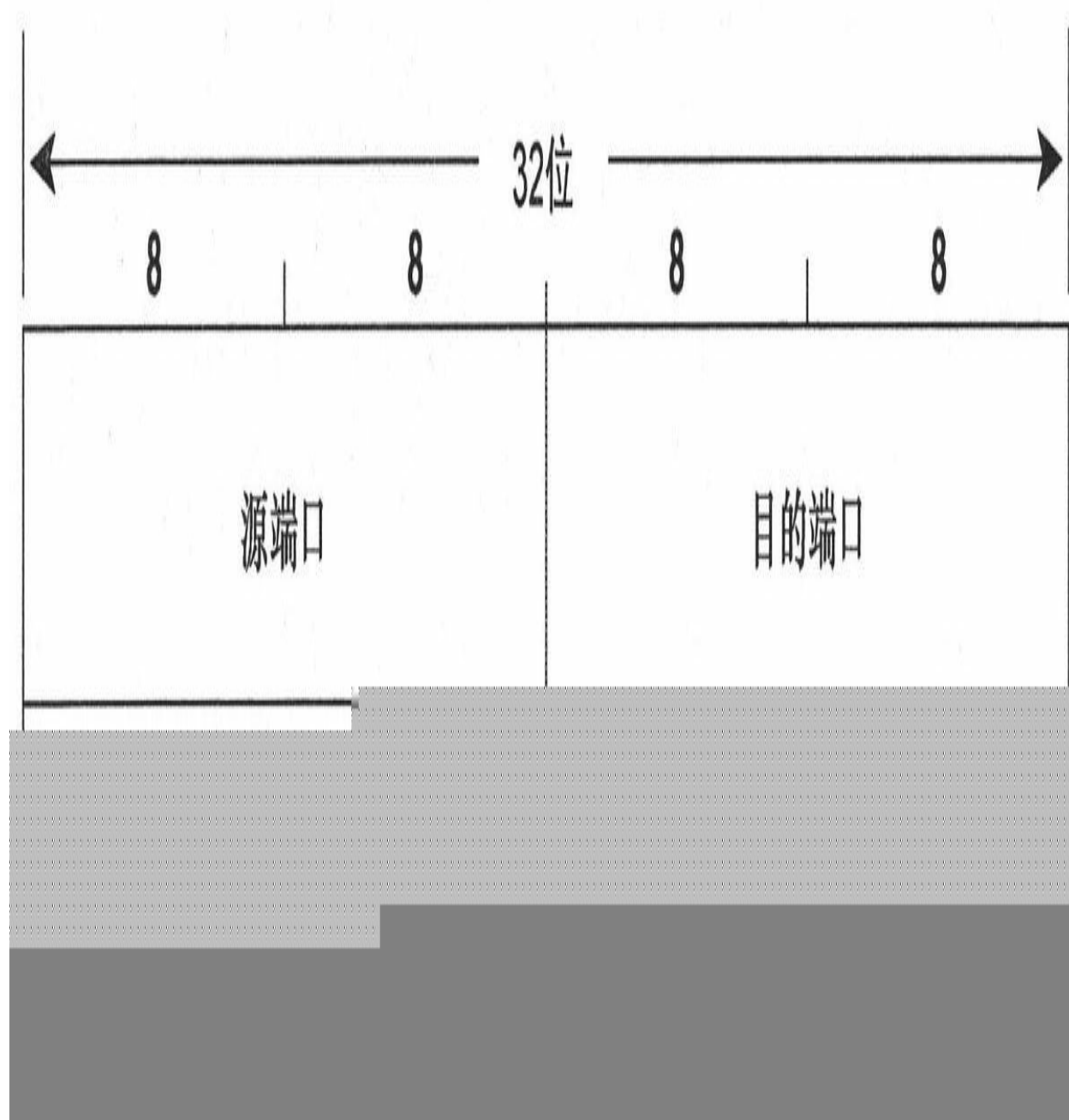


图2-9 路由器通告消息的格式

跳数限制（**Hop Limit**）—— 如果设定了跳数限制字段的值，那么链路上的节点将在它始发到链路上的所有数据包里都设置该值。如果这台路

由器没有指定跳数限制，那么该字段设置为全0。

M——是管理地址的配置标记。如果设置了该位，始发路由器就会利用DHCPv6协议来告诉链路上的主机使用有状态地址自动配置。如果不设置该位，链路上的主机应该使用无状态地址自动配置。地址自动配置将在本章后面的章节中讲述。

O——是其他的有状态配置标记。当设置该位时，始发路由器就会告诉所在链路上的主机使用DHCPv6协议来获取其他的链路信息。**M**标记和**O**标记可以一起使用。例如，不设置**M**标记但设置**O**标记，那么路由器将会告诉链路上的主机使用无状态地址自动配置，但对于其他的配置参数则不考虑DHCPv6服务器的存在。

路由器生存时间（Router Lifetime）——只有在始发路由器是一台缺省路由器时，该字段才设置为非0的值。如果是这种情况，则该字段指定为该缺省路由器的存活时间，以秒为单位，它的最大值为18.2h。

可达时间（Reachable Time）——是指用于实现NDP协议中邻居不可达性检测功能的字段。当一个节点确认它的邻居是可到达的后，可达时间会指定一个时间值，在该时间内，这个节点假定它的邻居是可达的，以毫秒为单位。

重传计时（Retransmit Timer）——用于实现NDP协议里有关地址解析和邻居不可达性检测功能的字段。它指定了重传的邻居请求消息之间的最小时间，以毫秒为单位。

在路由器通告消息的可选项字段里携带的可能选项包括以下内容：

- 发起路由器通告消息（RA）的接口的链路层地址。
- 路由器通告消息所在链路的MTU说明。
- 分配给链路的一个或多个前缀。该信息是无状态地址自动配置的基本信息，它告诉链路上的主机该链路的前缀信息。

如图2-10所示，图中说明了路由器请求消息的格式。路由器请求消息的ICMPv6类型值为133，而代码值为0。封装路由器请求消息的IPv6数据包的源地址，要么是始发该消息的接口所分配的IPv6地址，要么是一个

用“: : ”（全部为0）表示的未指定地址，这种情况出现在没有分配地址的情况（在发起的主机节点刚刚开始进行地址自动配置时会出现没有地址的情况）。目的地址是所有路由器多播地址（FF02::2）。

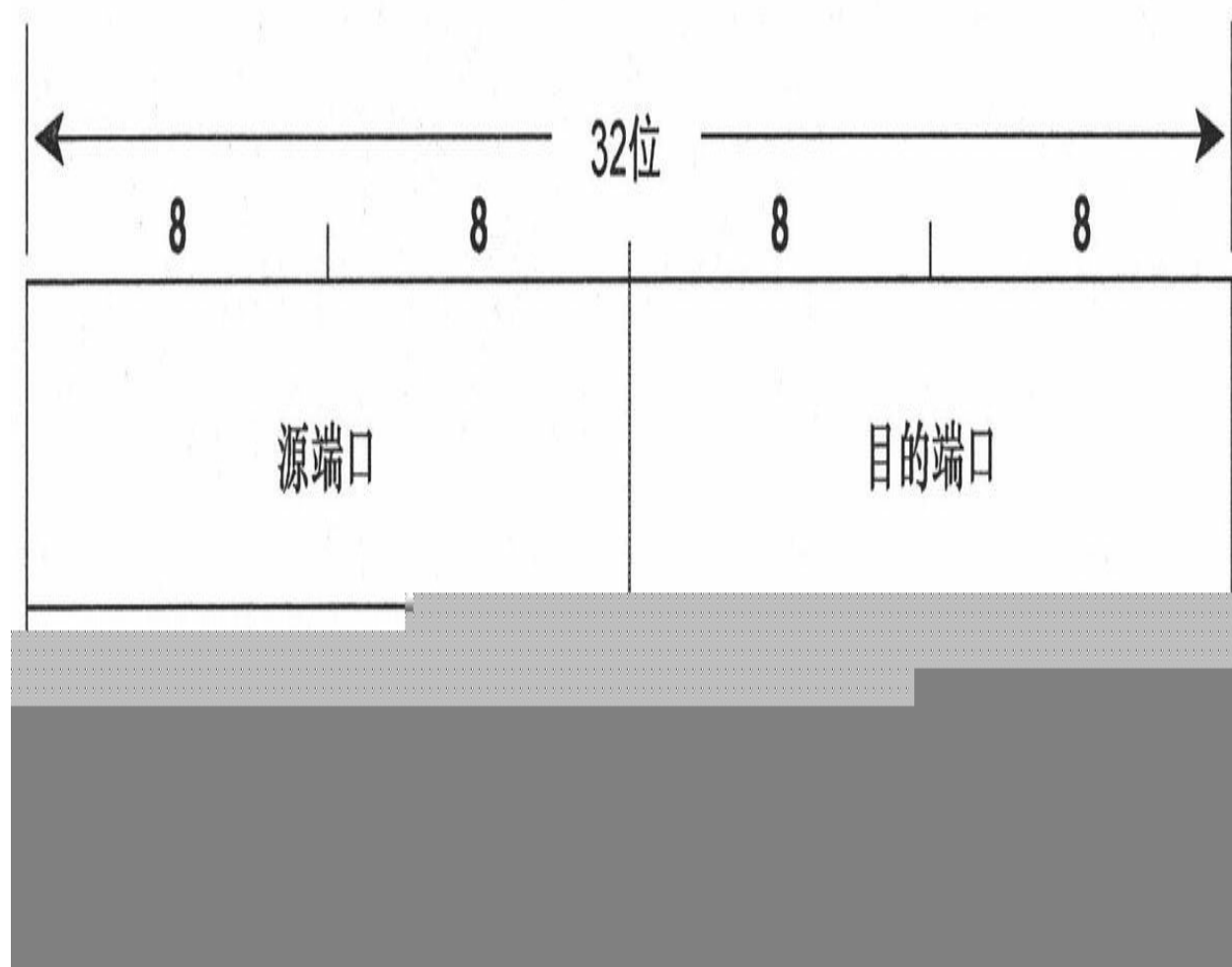


图2-10 路由器请求消息的格式

这里的可选项字段包括了始发该消息接口的链路层地址，如果可以知道的话。但是，如果封装该数据包的源地址是未指定的，那么就不一定包括源链路层地址，例如在地址自动配置期间始发主机正在请求路由器的时候。

在图2-11中，显示了邻居请求消息的格式。邻居请求消息的ICMPv6类型值为135，而代码值为0。封装邻居请求消息的IPv6数据包的源地址要么是始发该消息的接口所分配的IPv6地址，要么是一个用“::”（全部为0）表示的未指定地址，这种情况出现在为地址冲突检测而发送的邻居

请求消息的时候。目的地址是对应于目标地址的一个被请求节点的多播地址，或者就是目标地址。

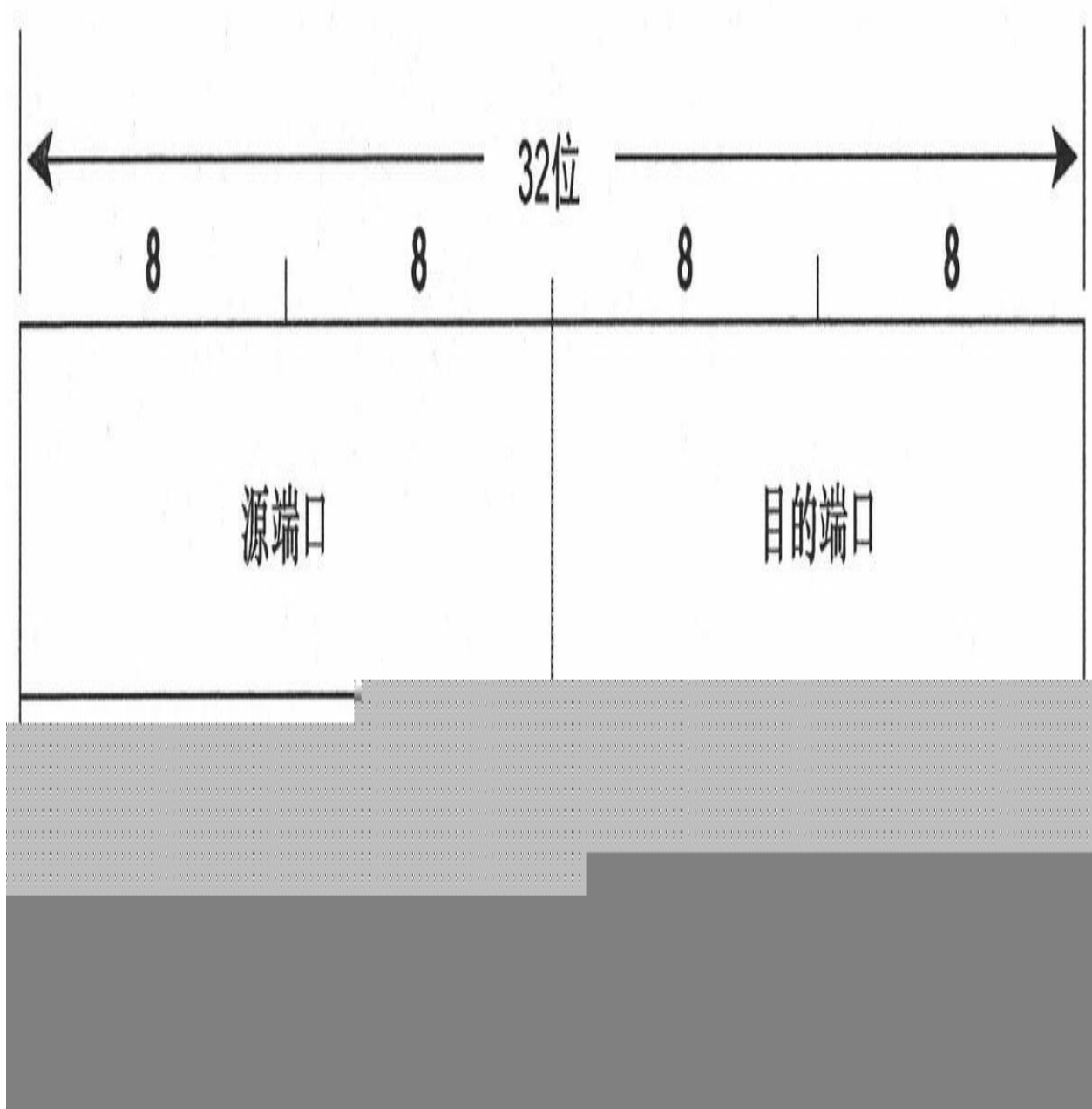


图2-11 邻居请求消息的格式

目标地址（**Target Address**）——是指请求目标的IPv6地址。目标地址永远不可能是一个多播地址。

邻居请求消息的可选项字段可以包括始发该请求消息的接口的链路层地

址。

如图2-12所示，图中显示了邻居通告消息数据包的格式。邻居通告消息数据包的ICMPv6类型值为136，而代码值同样为0。封装邻居通告消息的IPv6数据包的源地址总是由始发该消息的接口所分配的（或者自动分配的）IPv6地址。目的地址要么是包含该通告消息所要答复的邻居请求数据包的源地址，要么是所有节点的多播地址（FF02::1）。

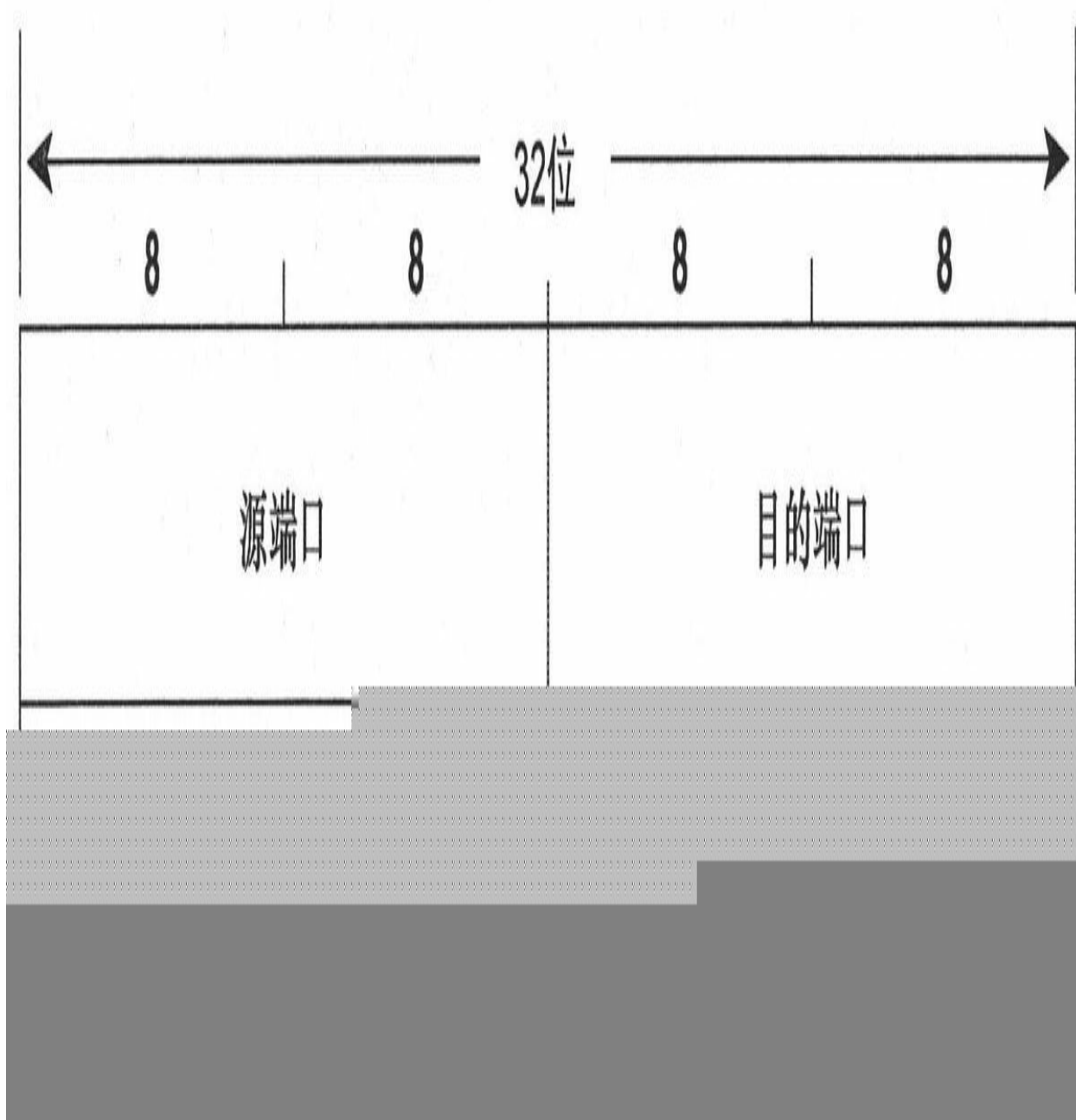


图2-12 邻居通告消息的格式

R——是路由器标记。如果设置了该位，表示始发该消息的节点是一台路由器。这个位字段用于在邻居可达性检测期间检查已经变为主机的路由器。

S——是一个请求标记。在为答复一个邻居请求消息而发送的邻居通告消息中设置该位。

O——过载标记。当设置该位时，它表示邻居通告消息中的信息超出了任何现有邻居的缓存条目，需要更新所缓存的链路层地址。当该位没有被设置时，说明邻居通告消息没有超出现有邻居的缓存条目。

目标地址——如果为了响应一个邻居请求而发送的邻居通告，那么目标地址就是邻居请求消息中的目标地址字段。如果邻居通告消息是未请求的（也就是说，发送通告始发节点的链路层地址的变化），那么目标地址就是始发节点的地址。

邻居通告消息的可选项字段可包含目标链路层地址——也就是说，是始发邻居通告消息节点的链路层地址。

如图2-13所示，图中显示了一个重定向消息的格式。重定向消息的ICMPv6类型值为137，而代码值同样为0。封装重定向消息的IPv6数据包的源地址总是始发该消息的接口的IPv6链路本地地址。而它的目的地址总是触发重定向的数据包的源地址。

目标地址——是指更好的第一跳地址——通常是链路上另一台路由器的链路本地地址。

目的地址——是指被重定向到目标地址的IPv6目的地址。

重定向消息的可选项字段包括目标节点的链路层地址，这和触发该重定向的报头大小相同，不会使它的数据包长度超过1280字节。

所有以上5种类型的消息中的可选项字段，无论包含什么信息，都是由一个或多个类型/长度/数值（TLV）这3个参数组合构成的。如图2-14所示，每一个TLV都是由一个8位的类型字段指定在数值字段中携带的信息的类型，一个8位的长度字段指定该数值字段的长度大小（用8位组作

为单位，即字节数），以及一个长度可变的数值字段组成。

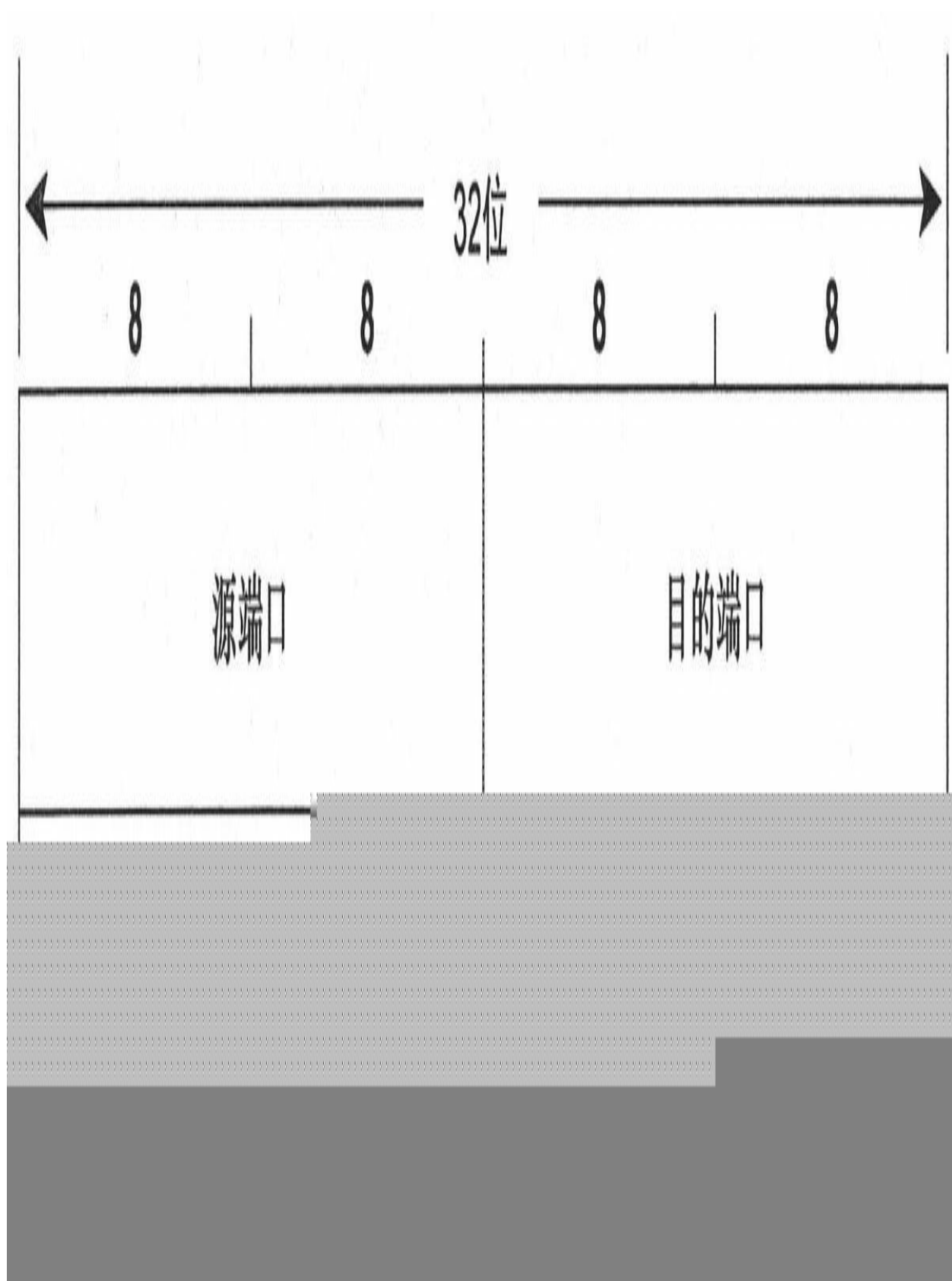


图2-13 重定向消息的格式

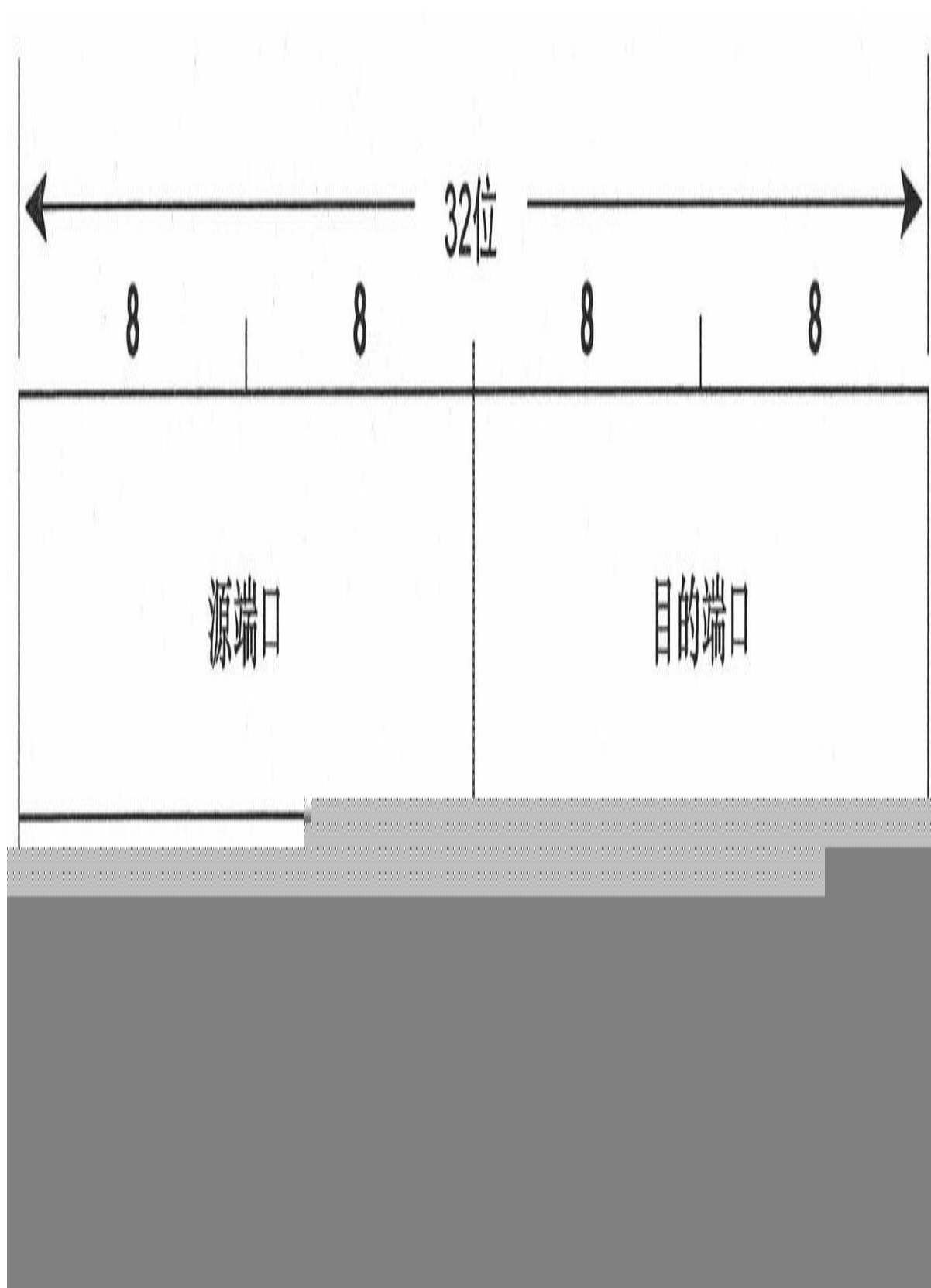


图2-14 用于NDP消息的可选项字段的TLV的格式

在表2-6中显示了可选的数值和它们相对应的类型号。本章并不讲述各个数值字段的格式，有关数值字段的详细描述，读者可以参考RFC2461。

表2-6 数值字段和相应的类型

类型	数值
1	源链路层地址
2	目标链路层地址
3	前缀信息
4	重定向的报头
5	MTU

2.5.2 路由器发现（Router Discovery）

路由器通过在相连的链路上周期性地发送路由器通告消息，表明它的存在，并通过路由器通告消息通告所配置的所有参数。大概用到路由器消息最多的链路就是广播链路了，例如以太网，链路上的主机可以收到路由器通告消息，因而可以学到有关链路的必要信息。

在RFC 2461中规定，路由器通告消息发送的周期是4~1800s之间，缺省值为600s。同时也规定了路由器通告消息通告的最小周期缺省为200s。这些通告将在最大值和最小值之间波动以防止链路上的同步。

一些未经请求的路由器通告消息的源地址是发送该消息的路由器接口的链路本地IPv6地址，而目的地址是所有节点的多播地址（FF02::1）。

在Cisco公司的路由器上，只要使用命令**ipv6 unicast-routing**使IPv6有效，Cisco路由器就可以在以太网和FDDI接口上自动地发送路由器通告消息。缺省的间隔值是200s，并且该值可以通过命令**ipv6 nd ra-interval**进行改变。缺省条件下，所传送的路由器通告消息的路由器生存时间是1800s，并可以通过命令**ipv6 nd ra-lifetime**进行改变。如果读者不希望链路上的某台路由器作为缺省路由器，就可以使用该命令将该路由器的生存时间设置为0。路由器通告消息的可达时间缺省为0（意味着是未指定的），并可以通过命令**ipv6 nd reachable-time**进行改变。缺省条件

下，重传计时字段设置为0ms（未指定的），并可以通过命令**ipv6 nd ns-interval** 进行改变。M标记与0标记字段的值可以分别通过命令**ipv6 nd managed-config-flag** 与**ipv6 nd other-config-flag** 来设定。如果读者根本不希望某个接口传送路由器通告消息，可以通过命令**ipv6 nd suppress-ra** 使其无效。

在缺省的情况下，Cisco路由器在路由器通告消息中包含了始发该消息的接口配置的所有IPv6前缀。可以通过命令**ipv6 nd prefix** 来控制前缀的通告，以及与这些前缀相关的参数。

刚刚连接到某个链路接口的主机需要等待一个路由器通告消息，以便能够发现链路上的路由器和学习到链路的参数，显然，要等待200s的时间显得过长了。因此，当链路上的一台主机开始激活时，就会发送一个路由器请求消息去请求立即得到一个路由器通告消息。这个路由器请求消息的源地址可以是未指定的地址 (::)，也可以是该主机的链路本地IPv6地址。它的目的地址则始终是所有路由器的多播地址 (FF02::2)。

当一台路由器收到一条路由器请求消息时，它就会发送（有一个5s的延时）一条路由器通告消息作为响应。如果触发某个路由器通告消息的路由器请求消息的源地址是一台主机的链路本地地址，那么这条路由器通告消息就会使用它的链路本地地址以单播方式被传送给该主机。如果路由器请求消息的源地址未指定，那么它所请求的路由器通告消息就会以多播方式被传送给所有的节点地址。

当一台主机收到一条路由器通告消息时，它就会将这台路由器添加到它的缺省路由器列表中（除非这条路由器通告消息指定它的路由器生存时间为0而不能作为缺省路由器使用）。如果一台主机在它的缺省路由器列表中具有多台路由器条目，那么主机就应该明确地给出如何选定一个缺省路由器的方法。它要么在整个缺省路由器列表中依次轮询，要么选择和保持单台路由器作为缺省路由。当出现与主机所选择的不同的缺省路由时，不管上述两种情况的哪一种，重定向功能对于用来更新主机的缺省路由都是重要的。

2.5.3 地址自动配置 (Address Autoconfiguration)

当一台IPv6的主机第一次连接到链路上时，它能够自我配置其自己的接口地址。这个过程的第一步是确定地址的64位接口ID部分。对于广播型

的接口（大多数主机是这种情况），使用一种称为MAC-to-EUI64转换法的机制。简单地讲，这个机制采用接口的48位介质存取 控制

（MAC）地址——通常认为它是全球惟一的，在这个MAC地址中间插入一个保留的16位数值0xFFFE，并把它的全局/本地（Universal/Local, U/L）位翻转设置为1（全局的，事实上是指IEEE指定），这样就把它转换成了一个64位的接口ID。

在图2-15中演示了这个转换过程，图中被转换的地址是0000:0B0A:2D51。为了更易于理解所发生的变化，在图中把MAC地址表示成二进制格式。第一步，把MAC地址从中间的位置分开成两个24位的对等部分，并在这两个24位对等部分之间插入0xFFFE。这样这个MAC地址就变成了64位的长度。接着，把原来MAC地址中的U/L位——始终是第7位——从0翻转为1。最后得到的结果就是一个有效的64位接口ID，即0200:0BFF:FE0A:2D51。

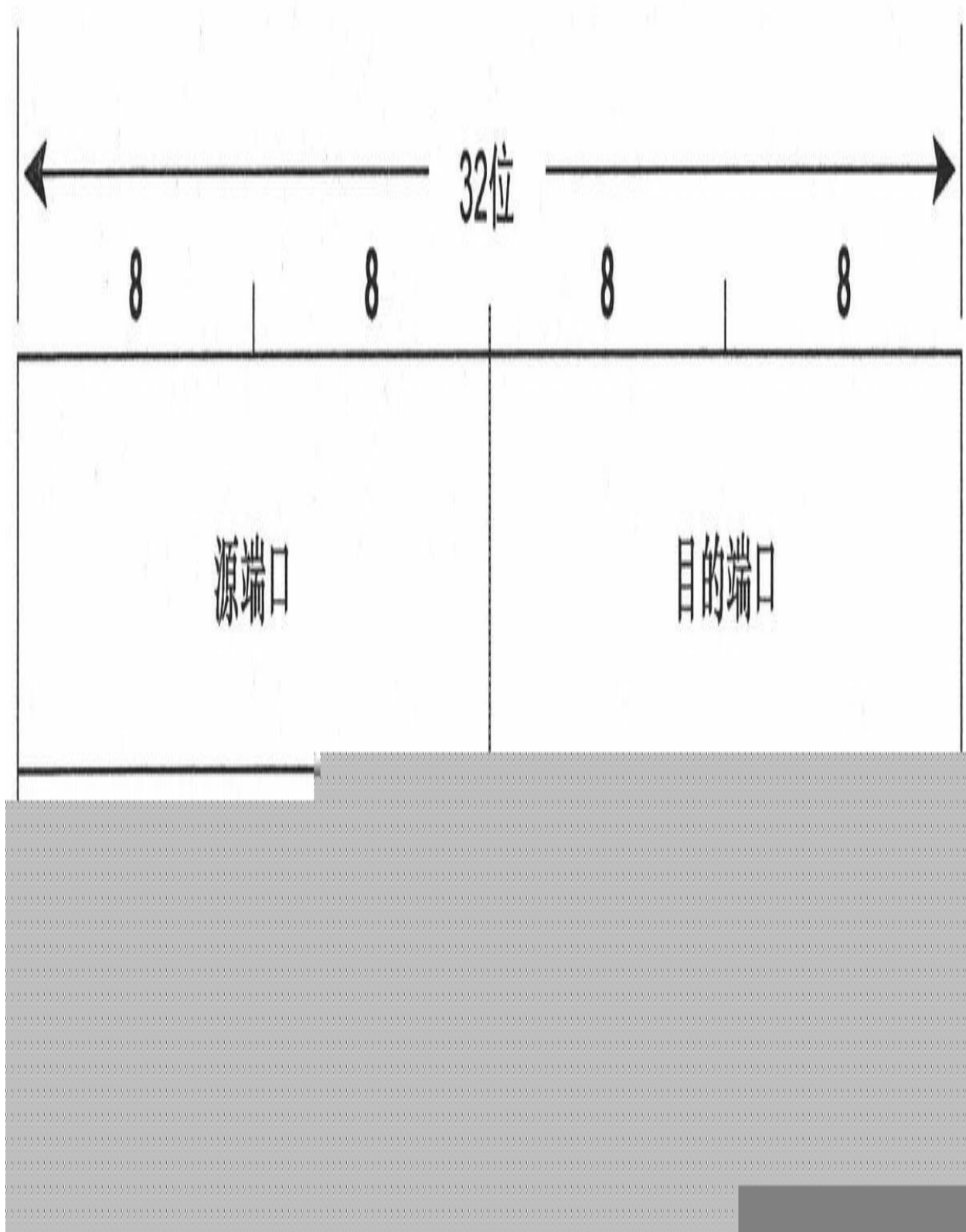
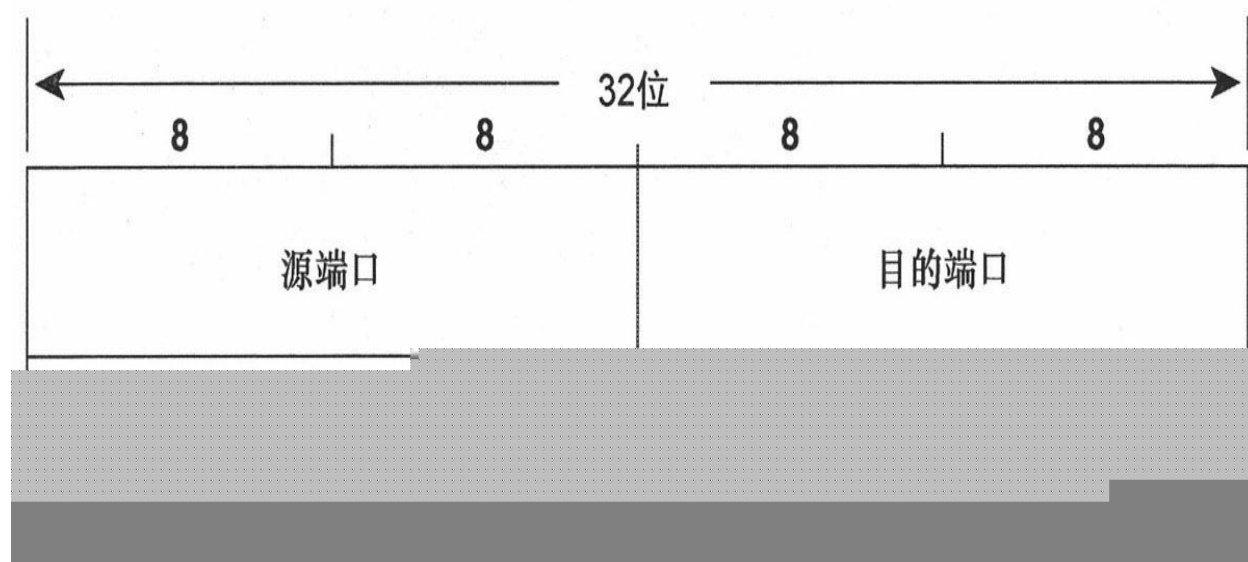


图2-15 MAC到EUI64转换法用来基于接口的48位MAC地址创建一个64

位的接口ID

当然，这个接口ID也仅仅是一个IPv6地址的一半，还需要一个64位的前缀。读者回想一下表2-1中链路本地前缀是一个保留的、公认数值0xFF80::/10。使用它作为一个完整的64位前缀（0xFF80::/64），再加上转换导出的接口ID，这样就构成了一个完整的IPv6地址，可以用来和同一链路上其他设备进行通信。例如，把链路本地前缀和图2-15中转换导出的接口ID组合在一起就得到了一个链路本地IPv6地址：

FF80::0200:0BFF:FE0A:2D51。下面显示了一个链路本地地址的例子，在该例中链路地址来自于一台Macintosh OS X主机的以太网接口“enl”。利用链路本地前缀FF80::/10和MAC到EUI64转换法，它的IPv6接口不用借助任何其他设备的帮助就可以导出自己的链路本地地址：



如果一台主机仅仅需要和所在链路上的设备通信，那么它自动配置的链路本地地址就已经足够了。但是如果主机需要和链路以外的设备进行通信，那么它就需要一个更大范围的地址——通常是一个全球IPv6地址。这可以通过两种途径获取该地址：有状态或无状态的地址自动配置。

如果一台主机使用有状态地址自动配置方式，它将会借助DHCPv6服务器来获取必要的地址信息。它要么根据预先的配置去查找DHCPv6服务器，要么由收到的可能设置了M标记的路由器通告消息来告诉它使用DHCPv6服务器。DHCPv6协议在RFC 3315中描述，它与IPv4的DHCP协议在最终结果上没有太多的不同。

更有趣的是无状态的自动配置。这是一个非常简单的过程，主机从它所收到的路由器通告消息中获取一个或多个链路前缀，然后加上它先前确定的接口ID，这样就得到了一个全球惟一的IPv6地址。举例来说，假设图2-15中的主机收到了一个路由器通告消息通告的前缀为3FFE:1104:404:1::/64，把这个前缀加上它的接口ID就构成了它的全球地址：3FFE:1104:404:1:0200:0BFF:FE0A:2D51。

2.5.4 地址冲突检测（Duplicate Address Detection）

虽然利用MAC地址转换导出一个接口ID的方法，大多数情况下通常可以保证在任何范围下得到一个惟一的地址，但是确保地址的惟一性无疑是明智的。所以不管一台设备是如何获取一个地址的，在使用这个地址之前都必须进行地址冲突检测。地址冲突检测不考虑地址是通过有状态或无状态配置方式获取的，还是人工静态配置获取的。这个规则只有一个例外情况就是任意播地址，因为定义的任意播地址可以出现在不止一台的设备上。在这里存在这样一个假设：如果一个链路本地地址具有通过MAC到EUI64转换法导出的接口ID，通过了地址冲突检测并被认为是惟一的，那么其他使用相同的接口ID的地址也被认为是惟一的，而不需要重新执行地址冲突检测。

已经获取一个新地址的节点会把这个新的地址归类为临时状态的地址。在地址冲突检测操作没有完成并确认链路上没有其他节点使用这个地址之前，该地址不能被使用。这个节点会发送一个把目标地址字段设置为该地址的邻居请求消息来验证。邻居请求消息的源地址是未指定的地址，而它的目的地址是被请求节点的多播地址。

一个被请求节点的多播地址由前缀FF02:0:0:0:0:1:FF00::/104前导，加上目标地址的最后24位组成。举例来说，对于图2-15中导出的接口ID，这里被请求节点的多播地址是FF02::1:FF0A:2D51。这样做的原因是如果一个节点自动配置了多个接口地址，它所有地址的最后24位都应该是相同的。因此，一个带有被请求节点多播地址的邻居请求消息应该匹配它所有的接口地址。更重要的是，使用被请求节点的多播地址可以确保在两个节点试图同时对同一个地址执行地址冲突检测操作的时候，它们可以互相检测到。

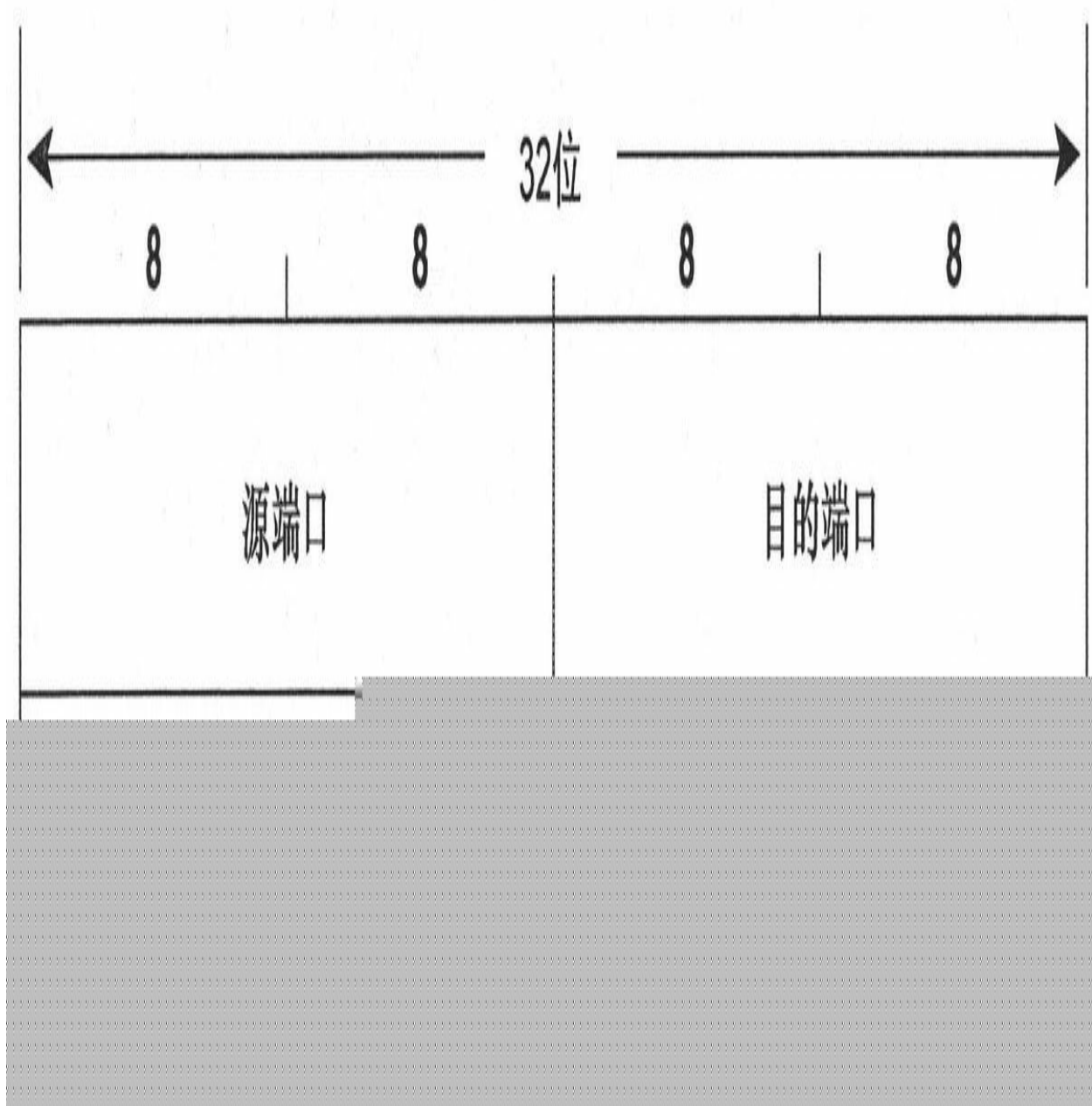
如果一个节点收到一个邻居请求消息，并且它的目标地址与该节点所分配的其中一个地址匹配，它就会发送一个目标地址和目的地址都为试探

地址的邻居通告消息。发起这个邻居请求消息的节点就会知道这个试探地址是冲突的，并且不能使用。

2.5.5 邻居地址解析 (Neighbor Address Resolution)

读者在第1章中已经了解到，当一个IPv4节点希望和本地链路上的另一个IPv4节点通信时，它必须首先发现目的节点的链路层地址（或称为数据链路地址）；然后这个地址被作为封装IP数据包到那个节点的数据帧的目的地址。举例来说，一个节点可能希望发送一个数据包到 `examplehost.com`，DNS的查询会返回地址 `3FFE:521:2400:15:211:24FF:FE23:334E`，发送端节点这时必须发现链路层地址用作本地链路帧的目的地址。正如前面章节所讨论的，IPv4协议使用ARP获取这个地址，而IPv6则使用NDP获取这个地址。

当一个节点检验由DNS返回的IPv6地址的前缀后，它就可以判断出目的节点是它所在的本地链路上的邻居，还是一个本地链路以外的节点并因此需要通过缺省路由器才能可达。如果是属于后一种情况，这个节点应该已经知道来自路由器通告消息的缺省路由器的链路层地址。但是，如果目的节点位于本地链路上，那么节点就会首先查找邻居缓存看一下是否已经学到该地址。IPv6协议中的邻居缓存与IPv4协议中的ARP缓存非常类似，它记录了网络层的地址和与之相关联的链路层地址。下面显示了一个微软WindowsXP操作系统主机的邻居缓存。这个邻居缓存保存了主机已知的IPv6地址和与它们相对应的链路层地址：



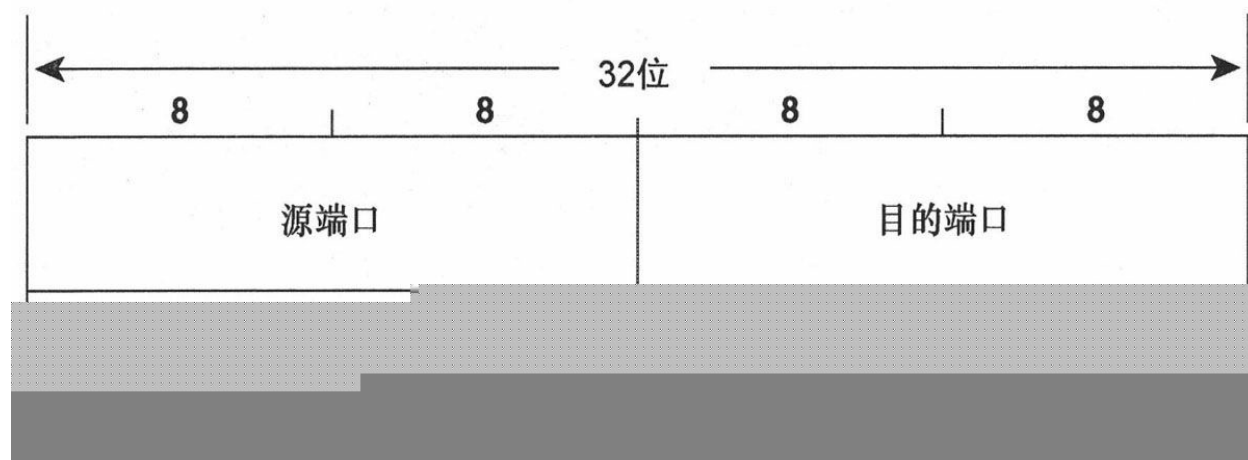
如果一个地址不在邻居缓存中，它也显示在表中但被标记成不完全的（Incomplete），这表示正在进行该地址的地址解析处理。这样，主机将发送一个邻居请求消息到与目标节点相关的被请求节点的多播地址。邻居请求消息应该包含源链路层可选项（类型1），这样被请求的节点就会了解到请求节点的链路层地址，并因此知道发送作为答复的邻居通告消息到哪一个节点。如果在路由器通告消息中包含了不为0的值，那么根据指定的时间间隔可以发送多个邻居请求消息。如果一个路由器通告消息中的重传计时字段的值为未指定的（0），那么邻居请求消息将

会每1000ms重传一次，直到收到一个邻居通告消息为止。如果发送了3个邻居请求消息后，还是没有收到来自被请求节点的邻居通告消息，那么邻居地址解析就会失败，并会为每一个排队等待传送到目前还未知的目的地的数据包返回一个类型1/代码3（目的地不可达/地址不可达）的ICMP消息。

假如被请求的节点存在并且邻居请求消息是有效的，那么它将答复一个邻居通告消息。这个邻居通告消息的目标地址字段设置为触发该邻居通告的那个邻居请求消息的目标地址字段的值。直到接收到回应的邻居通告消息，发起请求的节点才能够将目标节点的链路层地址添加到邻居缓存的条目中，并把该条目从不完全的（Incomplete）状态更改成可达的（Reachable）状态。

在Cisco路由器上，可以通过命令**show ipv6 neighbors** 来查看邻居缓存，参见示例2-1。

示例2-1 在Cisco路由器上，可以通过命令**show ipv6 neighbors**来显示邻居缓存



2.5.6 私有地址（Privacy Address）

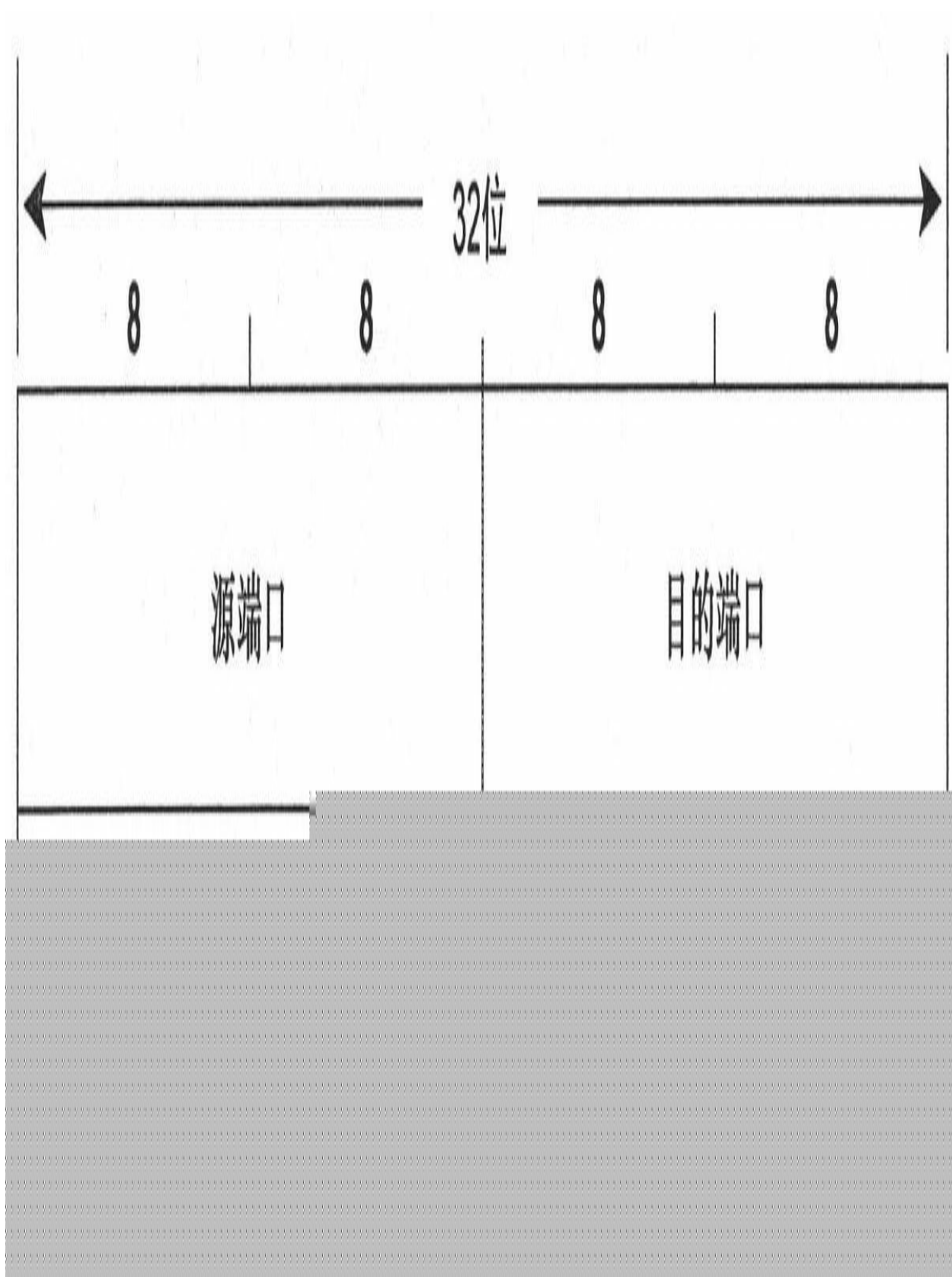
无状态地址自动配置在涉及到以下的一些情况时会产生一种安全问题：即使一台设备从一个子网移到另一个子网，或者从一个主网络转移到另一个主网络，它的接口ID也始终保持不变。如果它的接口ID保持不变，那么它能够被追踪。至少，这成为了一个私密的问题。例如，假设读者正在使用IPv6的地址和其所在公司的网络相连。记录和分析来自网络

某部分的数据包就能够通过不变的接口ID来识别读者的身份。而且，你的同事通过进一步分析前置在你的接口ID前的不同前缀，推断出任何时候你所在的位置：在公司工作，在家里，旅行，或者任何活动。更为严重的是，他们可以利用这些追踪以及保存有关你的活动和位置的记录，来进行犯罪交易。

在RFC 3041中，通过定义IPv6私有地址来解决该安全问题。私有地址就是通过利用伪随机数字算法生成的接口ID。接口ID大约一天变化一次（或根据所配置的周期变化），也会在节点获取一个新的IPv6地址时改变。这一点非常重要，也符合合理的保密要求。

当然，一个经常变化的地址对于可达性而言是不实际的。希望与你通信的节点，以及DNS服务器必须通过一个或一些静态地址了解你所在位置。因此，标准的无状态配置的IPv6地址保留了你的“公共”地址；任何一个需要向你发送数据包的节点都使用这个地址作为目的地址。但当你发送返回的数据包时，你使用的却是私有地址。这有点像你家里的主叫ID（Caller ID），但它会阻止你的号码显示在其他人的主叫ID上。你能够看到是谁正在呼叫你，但是，当你呼叫其他人的时候他们却不能看到你的号码。

下面显示了分配给一台微软Windows XP系统的地址。在这里，有两个公共的IPv6地址分配给这台机器的接口，读者可以看到它们的前缀虽然不同，但它们的由MAC到EUI64转换得到的接口ID却是相同的。可以通过插入到地址中间的0xFFFE来容易地识别出公共接口ID。但是，对于这两个公共地址，也有一个私有地址（Windows的标签是作为“anonymous”），这些私有地址是通过在随机生成的接口ID前加上路由器通告消息发现的IPv6前缀生成的。公共和私有的（这里称为“anonymous”）地址一起用来确保主机的隐匿并同时维持主机的可达性：



2.5.7 邻居不可到达性的检测

在前面的章节里讨论邻居地址解析时，已经提到了标记为Incomplete或Reachable的邻居缓存条目。事实上，一个邻居缓存条目可以具有以下5种状态：

- **Incomplete** ——该状态说明正在进行邻居地址的解析处理。一个邻居请求消息已经为该条目发送被请求节点的多播地址，但是还没有收到相应的邻居通告消息。
- **Reachable** 该状态说明目前地址已经被确认为是可到达的。“目前被确认”的意思是指在路由器通告消息中Reachable Time字段指定的时间内，已经收到了某些可达性的信息。如果在路由器通告消息里没有指定Reachable Time时间，那么将使用30s作为缺省的Reachable Time时间。
- **Stale** 该状态说明自从收到最新的有关到达目的地的可达性肯定的确认后，已经经历了Reachable Time所指定的时间。
- **Probe** ——该状态是指每经过Retransmit Time时间或每1000ms（如果没有指定Retransmit Time时间的话），节点就会通过向目的节点发送邻居请求消息来搜索可达性的确认。
- **Delay** ——当一个数据包发送到一个处于Stale状态的节点时，这个地址就进入到该状态。如果该地址在Delay状态等待了5s，并在这段时间内还没有收到有关可达性的确认，那么该状态就转换到Probe状态。这个状态是在节点发出搜索的邻居请求消息前，给上层协议一个确认可达性的机会的优化措施。

对于邻居可达性的确认，一般可通过以下两种方法来确认：

- 来自上层协议的“提示（Hints）”，例如像一个TCP消息的ACK等。
- 通过请求一个路由器通告消息或邻居通告消息得到有关对目的地址搜索的响应。这是一种必要的措施，因为对于某些上层协议，例如UDP协议，它本身并不对所传送消息的接收进行主动的确认。

邻居不可达性的检测不仅仅要确认从邻居到本地节点一方的可达性，而且还要确认从本地节点到邻居方向的可达性，确保双向的可达性。基于这个原因，一个未经请求的路由器通告消息或邻居通告消息，不能改变邻居缓存条目的状态到**Reachable**；所收到的消息仅仅可以表明从始发节点到本地节点的单向可达性。只有收到一个传输层消息的远程响应消息（例如一个TCP数据包的ACK），或者收到一个响应请求消息的路由器通告消息或邻居通告消息后，才能确认双向的可达性。

2.6 展 望

讲述本章和前面一章的目的是为了阐明基于两个版本的IP协议的基础概念。理解IP地址的基本概念和有关IP的基础知识可以作为下一步掌握IP路由选择知识的基础。本书后面的章节将开始研究一台路由器能够成功地和精确地转发数据包到达它的目的地所需要的信息。

2.7 复习题

1. 一个IPv6地址的长度是多少？
2. 怎样表示一个IPv6地址？
3. 用来压缩和简化IP地址表示的两条规则是什么？
4. 为什么在一个IPv6地址里使用多个双冒号是不允许的？
5. IPv6地址::<0和::<128有什么不同之处？
6. 在单播IPv6地址中，用来指定主机的部分是什么？它的长度是多少？
7. 一个单播IPv6地址的子网ID部分的长度是多少？
8. 假设一个IPv6地址的起始10位是FF80::- 9. 3FFE:204:100:90::1是什么地址类型？
- 10. 什么是任意播地址？
- 11. 什么是多播地址？
- 12. 一个IPv6报头的长度是多少？
- 13. 在IPv6报头中设置流标签字段的目的是什么？
- 14. 在IPv4报头中，什么字段和IPv6的下一报头字段相对应的？
- 15. 在IPv4报头中，什么字段和IPv6的跳数限制字段相对应的？
- 16. 在IPv6的下一报头字段中，有哪些方面像IPv4的协议号字段？又有哪些地方不同？
- 17. 怎样扩展报头以便使IPv6的数据包更加富有效率？

18. ICMPv6中下一报头的值是什么？
19. IPv4的分段和IPv6的分段有哪些重要的不同之处？
20. 用于邻居发现协议的5种ICMPv6消息是什么？
21. 在一个路由器通告消息中，M标记和O标记的用途是什么？
22. 在一个路由器通告消息中，可达时间字段的用途是什么？
23. 在一个路由器通告消息中，重传计时字段的用途是什么？
24. 在一个路由器通告消息中，如果它的路由器生存时间字段被设置为0代表什么意思？
25. 在一个邻居通告消息中，S标记的用途和作用是什么？
26. 有状态地址自动配置和无状态地址自动配置有什么不同？
27. MAC到EUI64转换使用哪两个步骤可以导出一个接口ID？
28. 在一台设备获取一个单播IPv6地址时，它必须执行地址冲突检测操作，但是有一个例外情况，这个例外情况是什么？
29. 前缀FF02:0:0:0:0:1:FF00::/104表示什么意思？
30. 在IPv6中，使用什么代替ARP和ARP缓存？
31. 什么是私有地址？
32. 在一个邻居缓存中，Incomplete状态的条目表示什么意思？
33. 在一个邻居缓存中，Probe状态的条目表示什么意思？
34. 邻居不可达性检测使用哪两种方法来确认一个邻居双向的可达性？

[\[1\]](#) 考虑到在IPv4地址分配时所没有预见到的情况——当时IPv4所拥有的43亿个地址被认为可以无限的提供给所有的实际需求，再也不会有人认

为IPv6协议所提供的几乎不可思议的巨大地址空间是不可耗尽的了。

[2] 在写本书时，共有5个RIR机构：RIPE（Région IP Européens）——服务于欧洲、中东地区和中亚地区；LACNIC（拉丁美洲和加勒比海Internet地址注册机构，Latin American and Caribbean Internet Address Registry）——服务于中美、南美以及加勒比海地区；ARIN（美国Internet编号注册机构，American Registry for Internet Numbers）——服务于北美地区和部分加勒比海地区；AfrinIC——服务于非洲地区；而APNIC（亚太地区网络信息中心，Asia Pacific Network Information Centre）——则服务于亚洲和太平洋地区的国家。

[3] 路由器在到达同一个目的地的多条路由中选择路由的方法将在第4章中讲述。

本章包括以下主题：

- 路由表；
- 配置静态路由；
- 静态路由故障诊断。

第3章

静态路由

第1章中有一点非常重要，就是在OSI模型定义中，数据链路层/物理层和传输层/网络层执行的任务非常相似：它们都提供了数据传输的手段，即沿某条路径将数据从源点发往目的地的方法。不同之处在于，数据链路层/物理层提供跨物理路径的通信服务，而传输层/网络层则提供跨由一连串的数据链路组成的逻辑路径或虚拟路径的通信服务。

此外，第1章中还阐述了沿物理路径进行通信，必须获得有关数据链路标识和数据封装的信息，并且这些信息要保存在数据库中，如ARP高速缓冲。同样，传输层/网络层也需要获取和保存所涉及到的相关信息。有区别的是，这些信息被保存在路由表中，路由表又叫路由选择信息库（RIB）。

本章所要研究的内容包括：需要何种信息路由数据包，怎样在路由表中保存这些信息，如何向数据库中输入这些信息，以及通过向正确的路由器的路由表中输入正确的信息来建立一个可路由网络的相关技术。

3.1 路由表

为了理解路由表中的信息种类，我们需要先考虑数据包到达路由器接口时会发生什么，这是非常有用的。首先，路由器会检查数据帧目标地址字段中的数据链路标识。如果它包含了路由器接口标识符或广播标识符，那么路由器将从帧中剥离出数据包并传递给网络层。在网络层，路由器将检查数据包的目标地址。如果目标地址是路由器接口的IP地址或是所有主机的广播地址，那么需要进一步检查数据包的协议字段，然后再把被封装的数据发送给适当的内部进程。[\[1\]](#)

除此之外，所有其他目标地址都需要进行路由选择。这里的目标地址可能是另一个网络上的主机地址，该网络或者与路由器相连（包括与那个网络相连接的路由器接口），或者不直接连接到路由器上。目标地址还可能是一个定向的广播地址，这种地址有明确的网络地址或子网地址并且主机位全部为1。这些地址也是可以路由的。

如果数据包是可以被路由的，那么路由器将会查找路由表获得一个正确的路径。在数据库中的每个路由表项最少必须包括下面两个项目：

- 目标地址 ——这是路由器可以到达的网络地址。正像本章所解释的，路由器可能会有多条路径到达相同的地址或是相同主网IP地址下的一组等长或变长的子网。
- 指向目标的指针 ——指针不是指向路由器的直连目标网络就是指向直连网络内的另一台路由器地址，或者是到这个链路的本地接口。更接近目标网络一跳的路由器叫下一跳（next hop）路由器。

路由器将会尽量地进行最精确的匹配。[\[2\]](#)按精确程度递减的顺序，可选地址排列如下：

- 主机地址（主机路径）；
- 子网；
- 一组子网（一条汇总路由）；
- 主网号；

- 一组主网号（超网）；
- 缺省地址。

本章将提供有关前4类的例子。超网将会在第6章中讨论。缺省地址是最不明确的地址，只有当所有匹配都失败时才被使用。缺省地址会在第12章中讨论。

如果数据包的目标地址不能匹配到任何一条路由表项，那么数据包将被丢弃，同时一个“目标网络不可达”的ICMP消息将会被发送给源地址。

如图3-1所示，这是一个简单的网络，图中给出了每台路由器需要的路由表项。这里最重要的是这些路由表将如何作为一个整体运行并能准确高效地传输数据包。路由表的网络栏列出了路由器可达的网络地址。指向目标网络的指针在下一跳栏中。

在图3-1中，如果路由器Carroll收到一个源地址为10.1.1.97、目标地址为10.1.7.35的数据包，路由表查询的结果是：目标地址的最优匹配是子网10.1.7.0，可以从SO接口出站经下一跳地址10.1.2.2去往目的地。数据包被发送给路由器Dahl，Dahl查找自己的路由表后发现数据包应该从S1接口出站经下一跳10.1.4.2去往目标网络10.1.7.0。此过程将一直持续到数据包到达路由器Baum。当Baum在接口SO接收到数据包时，Baum通过查找路由器，发现目的地是连接在端口E0的一个直连网络。最终结束路由选择过程，数据包被传递给以太网链路上的主机10.1.7.35。

上面说明的路由选择过程是假设路由器可以将下一跳地址同它的接口进行匹配。例如，路由器Dahl必须知道通过接口S1可以到达Lewis的地址10.1.4.2。首先Dahl从分配给接口S1的IP地址和子网掩码可以知道子网10.1.4.0直接连接在接口S1上；那么Dahl就可以知道10.1.4.2是子网10.1.4.0的成员，而且一定被连接到该子网上。

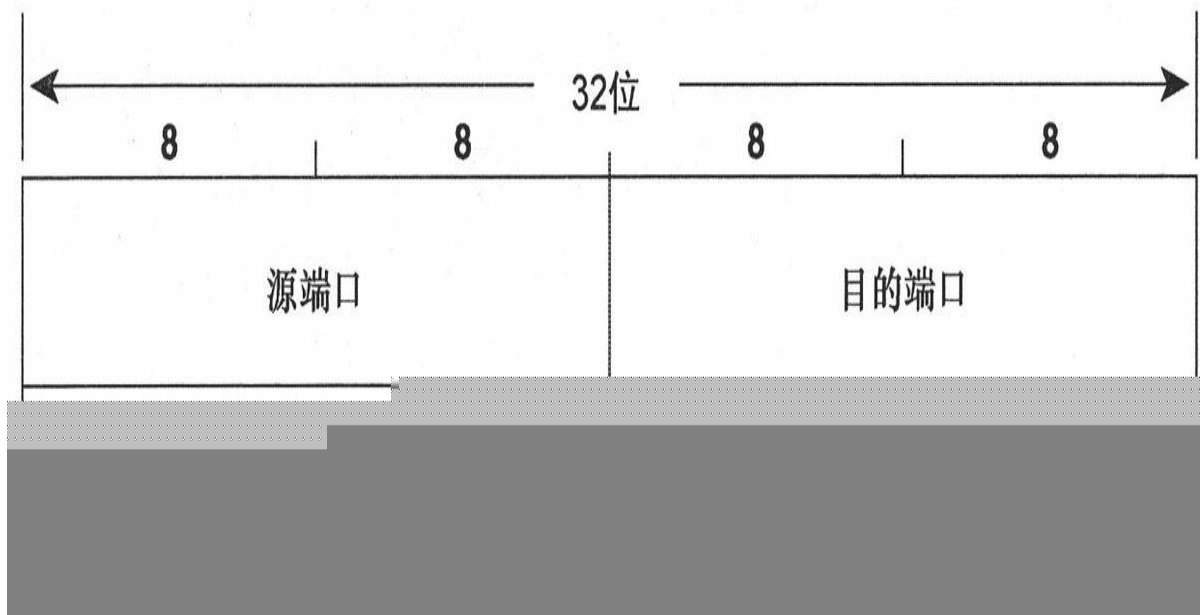


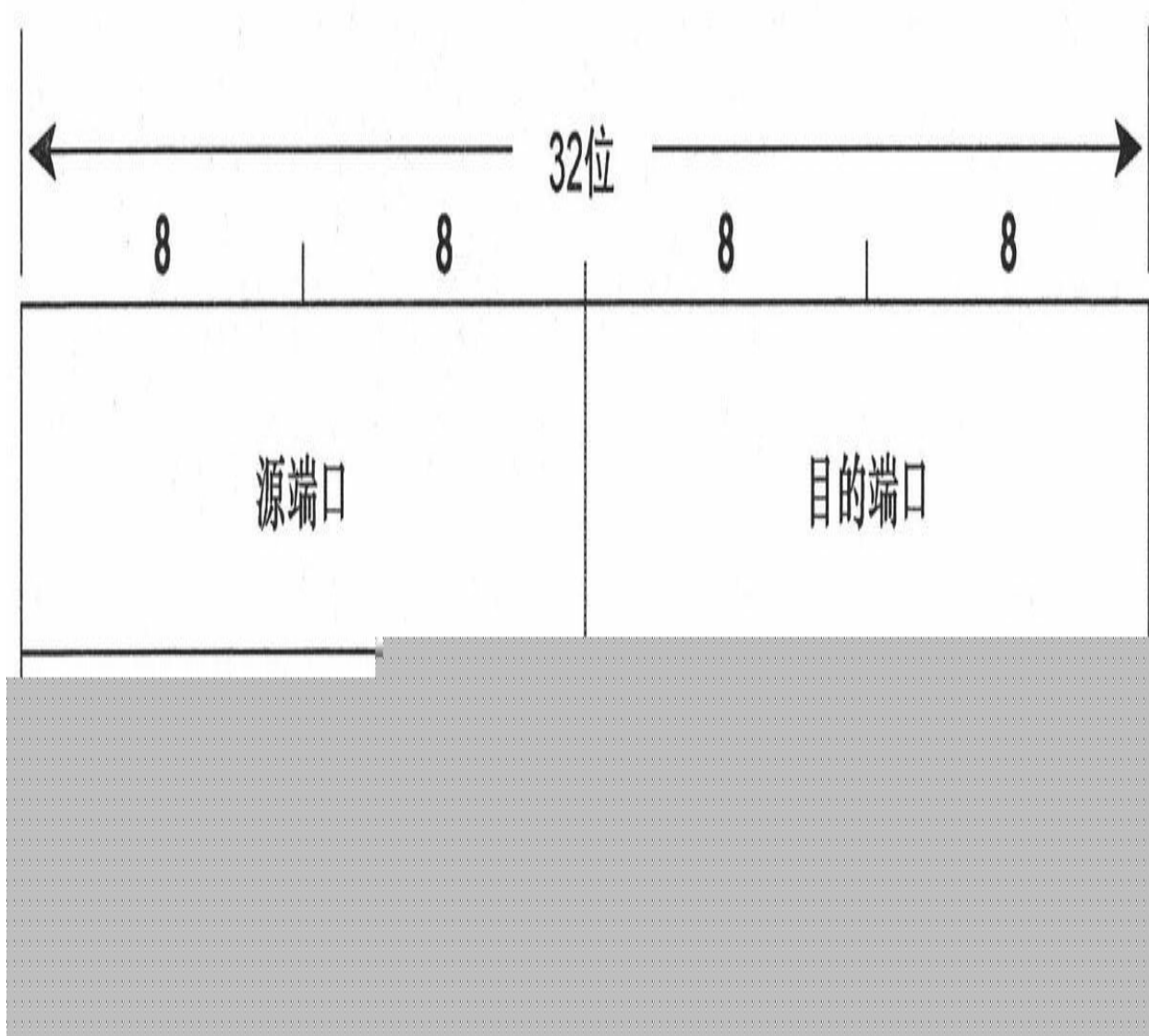
图3-1 每一个路由表项目需要的信息至少应该包括目标网络地址和指向目标网络地址的指针

注意，为了正确地进行数据包交换，每台路由器都必须保持信息的一致性和准确性。例如，在图3-1中，路由器Dahl的路由表中丢失了关于网络10.1.1.0的表项。从10.1.1.97到10.1.7.35的数据包将被传送，但是当10.1.7.35向10.1.1.97回复数据包时，数据包从Baum到Lewis再到Dahl传递。Dahl查找路由表后发现没有关于子网10.1.1.0的路由表项，因此丢弃此数据包，同时Dahl向主机10.1.7.35发送目标网络不可达的ICMP信息。

示例3-1给出了图3-1中路由器Lewis的路由表。在Cisco路由器中查看路由表的IOS命令是**show ip route**。

检查数据库的内容并把它与图3-1中路由器Lewis的一般路由表相比较。可以看到，表最上方的关键字是对路由表左侧的一列字母的解释。这些字母指明了每个路由表项是如何学习到的。在示例3-1中，标记为C的路由表示直连网络，标记为S的路由表示静态路由。声明“gateway of last resort is not set”指的是缺省路由。

示例3-1 图3-1中路由器Lewis的路由表



表头有一句声明主网络地址10.0.0.0有7个已知子网，掩码为24位。在7个路由表项中，每一个都给出了目标子网。对于不是直连网络的表项——数据包必须转发到下一跳路由器——置于括号内的元组指明了路由的[管理距离/度量]。管理距离将会在本章后面介绍，在第11章中还将详细讨论。

度量是通过优先权评价路由的一种手段，度量越低，路径越短，也就是该路径更理想。在第4章中将会详细讨论度量。注意，在示例3-1中静态路由的度量为0。最后，路由表还给出了下一跳路由器的接口地址或连接直连目标网络的接口。

3.2 配置静态路由

路由表可以用下面3种方式之一获取信息：

- 基于路由器的直连子网；
- 以静态路由表项的方式手工输入信息；
- 通过某种自动信息发现和共享系统（动态路由选择协议）自动地获取信息。

本书将花大量篇幅讲述动态IP路由选择协议，这里讨论一下静态路由配置有助于读者理解后继章节。

除了上述目的外，在某些场合，人们宁愿选用静态路由选择，而不是动态路由选择。对于任何过程而言，自动化程度越高，可控程度就越差。虽然动态（自动）路由选择要求更少的人为干涉，但静态路由选择允许在网络的路由选择行为上实施非常精确的控制。然而为此付出的代价是，每当网络拓扑结构发生变化时都需要重新进行手工配置。

3.2.1 案例研究：简单IPv4静态路由

如图3-2所示，网络有4台路由器和6个数据链路。注意，网络10.0.0.0的几个子网是不连续的——有一个不属于10.0.0.0的子网（在Tigger-to-Piglet链路路上的192.168.1.192）将子网10.1.0.0与10.0.0.0的其他子网分离开来。10.0.0.0子网是可变长子网——在整个网络中子网掩码长度不一致：子网10.1.0.0有16位子网掩码，而10.4.0.0是24位。最后要说的是，路由器Pooh的以太网链路路上的子网地址是一个全0子网。在后面章节可以看到这种编址方式对于更简单的有类别路由选择协议，如RIP和IGRP，将会产生一些问题；而静态路由则工作得很正常。

网络的静态路由选择过程共有3步：

步骤1： 为网络中的每个数据链路确定子网或网络地址。

步骤2： 为每台路由器标识所有非直连的数据链路。

步骤3： 为每台路由器写出关于每个非直连地址的路由语句。

这里没有必要写出有关直连数据链路的路由描述，因为在路由器接口上配置的地址和掩码可以使这些直连网络被记录在路由表中。

例如，在图3-2中的6个子网是：

- 10.1.0.0/16
- 10.4.6.0/24
- 10.4.7.0/24
- 192.168.1.192/27
- 192.168.1.64/27
- 192.168.1.0/27

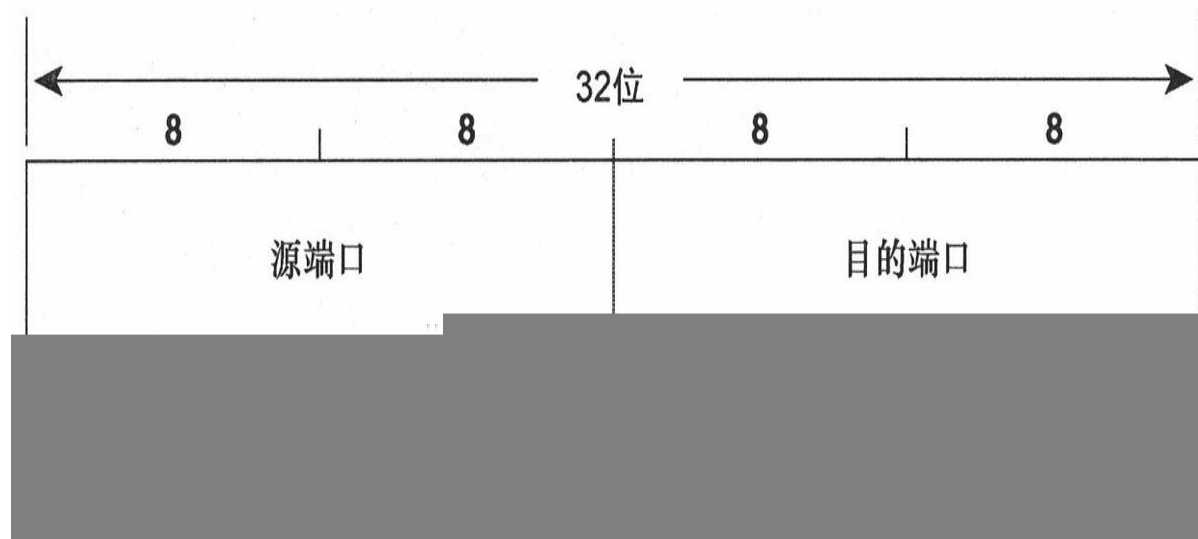


图3-2 路由选择协议RIP和IGRP对于含有非连续和可变长子网的网络来说不能进行正常的路由，而静态路由没有问题

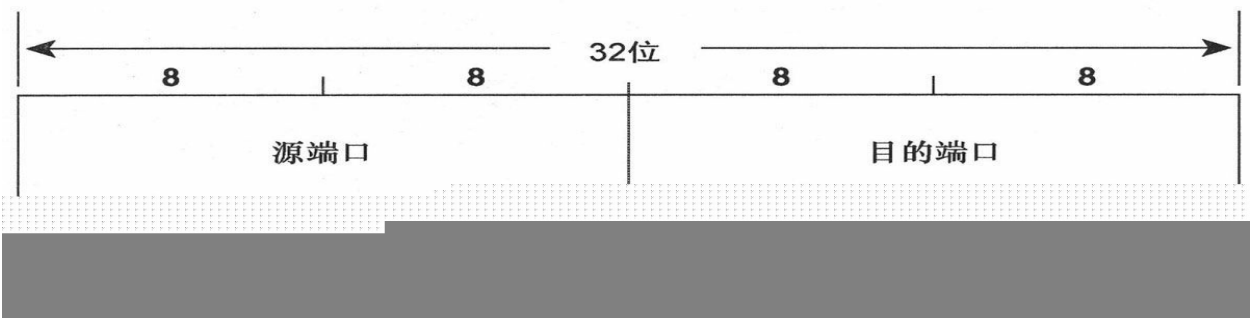
为了在路由器Piglet上配置静态路由，将那些非直连的子网标识如下：

- 10.4.6.0/24

- 10.4.7.0/24
- 192.168.1.64/27
- 192.168.1.0/27

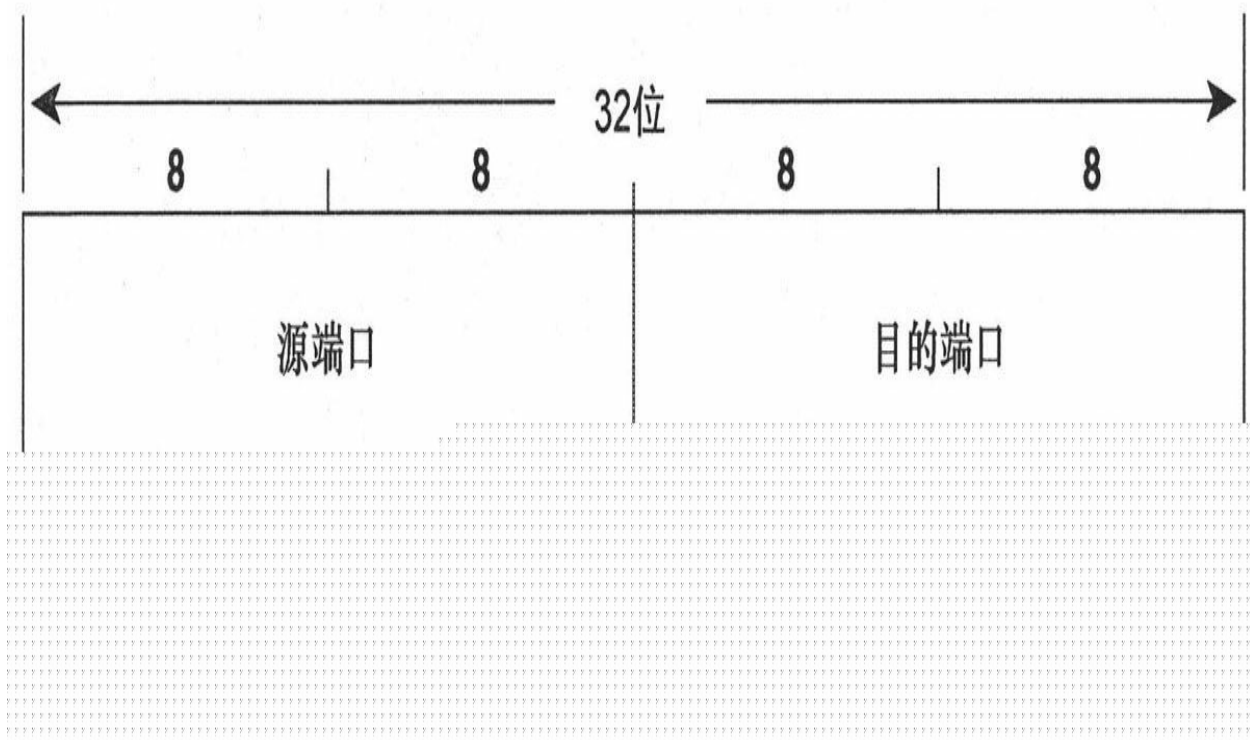
对于静态路由来说，这些子网必须被记录下来。在路由器Piglet上配置静态路由[\[3\]](#)的命令见示例3-2。

示例3-2 配置Piglet的静态路由表项



对其他3台路由器也执行相同的步骤，示例3-3给出了其他3台路由器的具体路由配置表项。

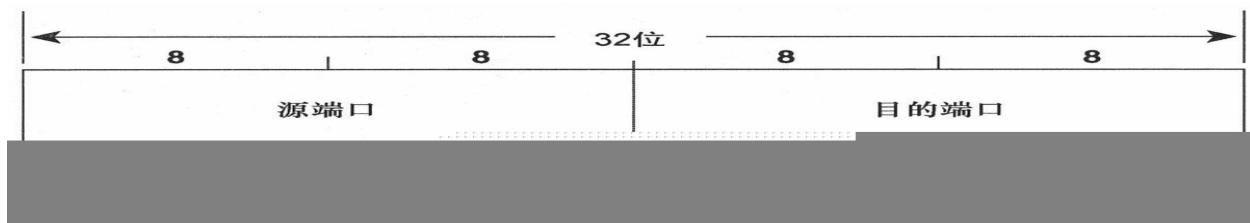
示例3-3 路由器Pooh、Tigger和Eeyore的路由配置



如果读者记住每条命令描述一个路由表项的话，路由配置命令本身是很容易阅读的。命令 `ip route` 用于 IPv4，它后面跟着的是将要被输入到路由表中的地址、确定地址网络号的掩码及下一跳路由器的链接口地址。

配置 IPv4 静态路由还可以选择另一种命令，这种命令用出站接口代替下一跳路由器的接口地址，其中通过出站接口可以到达目标网络。例如，如示例 3-4 所示，可以这样配置路由器 Tigger 的路由表：

示例 3-4 另一种配置 Tigger 的路由表项的方法

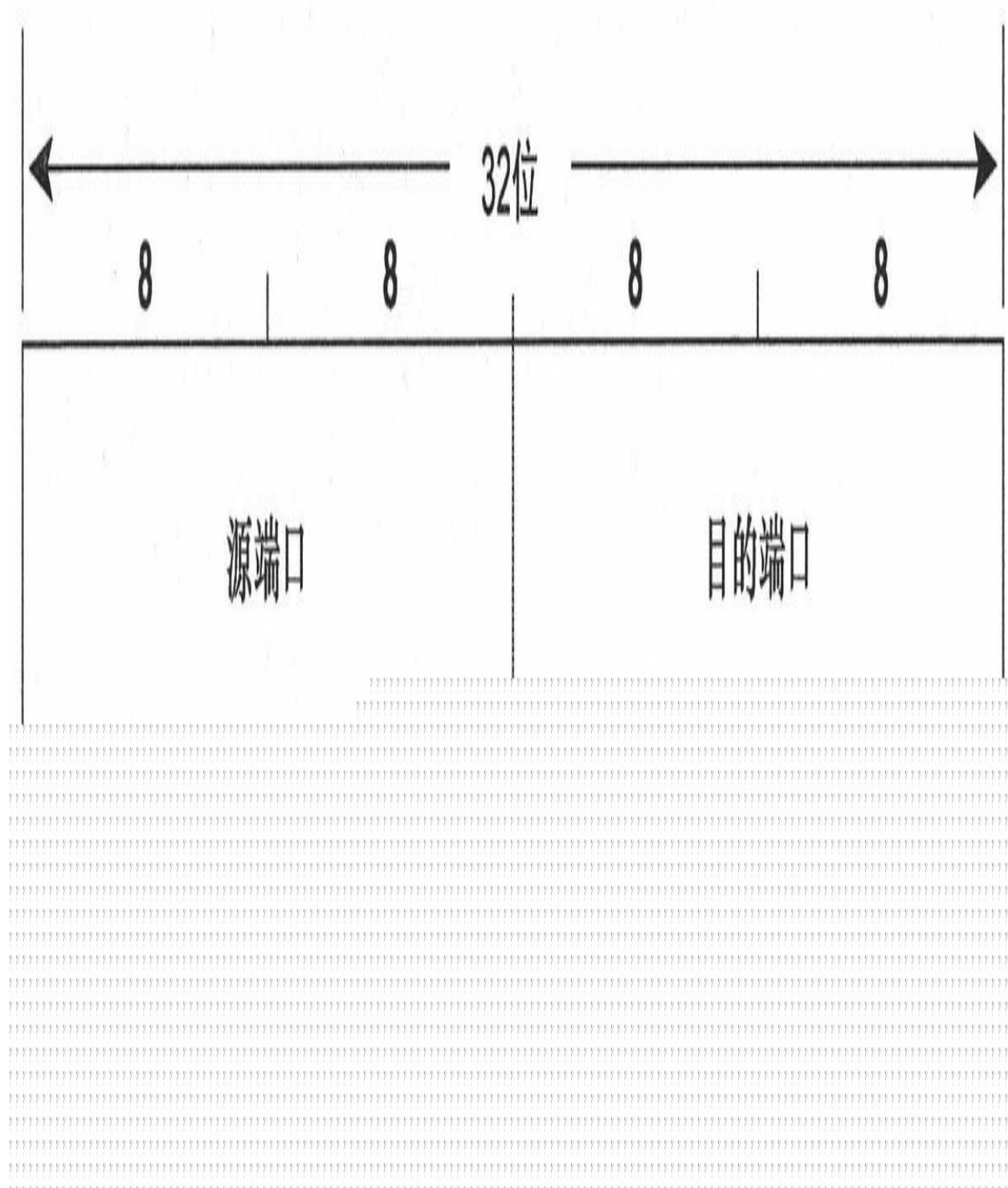


在配置静态路由时，必须满足以下条件。首先 IP 路由选择必须启动，如果使用下一跳地址，那么该地址必须可达；其次出站接口必须配置 IP 地址，接口必须正常工作。

示例 3-5 比较了两种配置方法所产生的路由表。注意，这里引入了一个

错误，所有用静态路由指明的网络，如果静态路由参照出站接口，那么它们将被作为直连网络输入到路由表中。这里所涉及到路由重新分配的问题将在第11章中讨论。

示例3-5 上面的路由表是指向下一跳路由器的静态路由表项产生的结果，下面是指向出站接口到达目标网络的静态路由表项产生的路由表
[\[4\]](#)



在示例3-5中，令人感兴趣的一点是，在10.0.0.0子网标题中指明了网络中使用了可变长子网掩码。可变长子网掩码（VLSM）是一种很有用的工具，我们会在第6章中讨论VLSM。

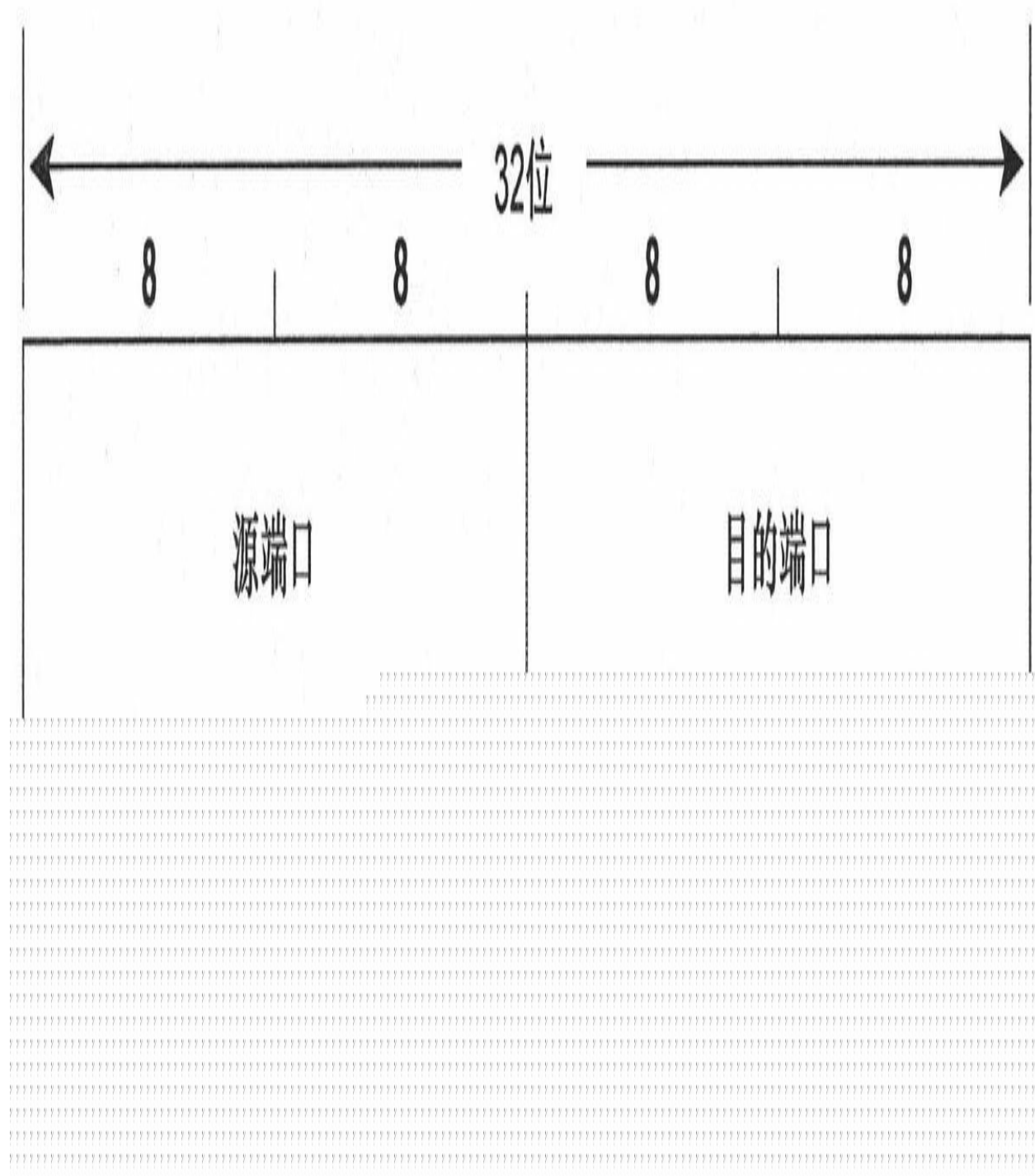
配置静态路由的第三种选择是联合使用出站接口和下一跳地址，即下一跳地址加上指定的出站接口。如果出站接口失效，即使下一跳地址通过替代路由递归可达，路由依然会被删除。这样可以把与下一跳地址相关的出站接口查询减到最小，同时使相应的路由表项不再是直连网络，而是距离为1的静态路由。

如果直接把静态路由指向一个广播型出站接口，而不使用下一跳地址，可能会导致广播网络上出现过多的流量，而且还可能耗尽路由器的内存。例如，如果我们在路由器Tigger上输入命令**ip route 10.1.0.0 255.255.0.0**，路由器会认为10.1.0.0是直连网络。当路由器向10.1.0.0/16中的目标主机转发数据包时，路由器会发送ARP请求以便获取目标主机的MAC地址。如果广播网络上的一台路由器（ARP代理）代表网络10.1.0.0回复ARP响应，那么不管数据包的目标地址是否有效，每次到达都会触发一个ARP请求和响应，并需要路由器配备大容量的**ARP** 高速缓冲。例如**ip route 10.1.0.0 255.255.0.0 E0 192.168.1.194**，如果在静态路由表项后面附加下一跳地址，那么路由器就不再认为目标网络是直连网络。这时，ARP的请求对象只可能是下一跳地址，而且仅当第一个去往10.1.0.0的数据包才会触发ARP请求。

指定出站接口和下一跳地址可以最小化与下一跳地址关联的出站接口查询，并且把广播网络上的流量减到最小。

如示例3-6所示，当联合使用出站接口和下一跳地址时，静态路由表项会有所不同。

示例3-6 在静态路由中指定出站接口，而不指定下一跳地址将会在广播网络上产生过多流量



第一组路由表和ARP高速缓冲没有使用下一跳地址，网络10.1.0.0是直连网络，并且有多个ARP高速缓冲表项与之相关联，其中硬件地址相同，且都是路由器192.168.1.194的MAC地址。该路由器代替网络10.1.0.0中的所有主机回复ARP请求。在第1章中曾经讲述过，ARP代理功能在IOS缺省状态下是打开的。

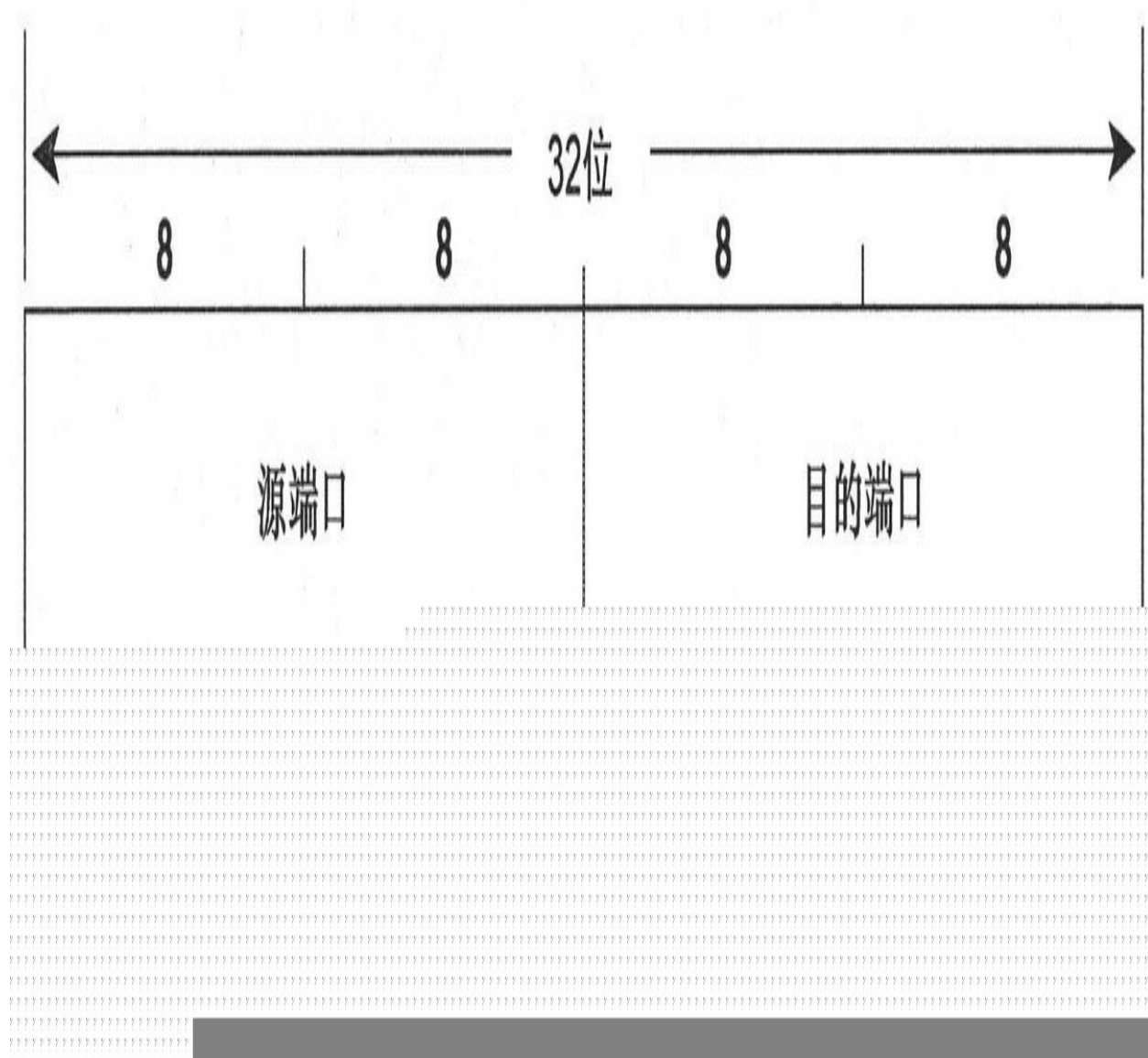
第二组表显示同时使用了出站接口和下一跳地址。注意，网络不再被直连连接了，而是通过192.168.1.194到达，出站接口是E0。ARP高速缓冲中没有关于10.1.0.0网络的表项，仅有实际存在的直连网络地址，其中包含192.168.1.194的。

3.2.2 案例研究：简单IPv6静态路由

IPv6静态路由的配置方法和IPv4基本相同，惟一不同的是IPv4网络掩码使用点分十进制形式，而IPv6使用目标网络的前缀长度。不像IPv4，IPv6路由选择缺省情况下是关闭的，所以在输入IPv6静态路由前，必须使用命令**ipv6 unicast-routing**启动IPv6路由选择。同IPv4一样，在向路由表中添加IPv6路由选择之前，出站接口必须有效，并且接口上已配置好一个IPv6地址。创建静态路由的命令是**ipv6 route**，该命令后面跟随的参数是目标网络、前缀长度（单位是比特）和下一跳路由器地址或去往目标网络的出站接口。

当我们需要确定静态路由表项中下一跳地址时，详细的网络图表会有所帮会，但是地址接口ID部分的动态特性又使得图表的信息容易变得过。在为IPv6网络分配地址时，要想预先指定下一跳地址，就必须手工指定接口ID，而不能使用自动构建的EUI-64格式的地址。如果数据链路上接口已经配置了EUI-64的接口ID，那么就只能指定地址的前64位。路由器将基于MAC地址确定后64位。如果路由器被新路由器替换，那么相应的IPv6地址也不同（前64位保持相同，后64位将不同）。为了标识邻接路由器的128位IPv6地址，可使用Cisco发现协议（CDP）的统计信息。CDP可以给出关于邻接路由器的信息，例如路由器的主机名、路由器类型、IOS和链路远端的IP地址。命令**show cdp**的显示格式如示例3-7所示。

示例3-7 Cisco发现协议可以给出大量关于邻居设备的信息



示例3-7给出了大量关于邻居设备的信息，包括路由器类型、IOS、主机名和IP地址。在命令中使用关键字**detail** 可以获取所有显示信息。

另一种确定链路IPv6地址的方法是使用**show ipv6 interface** 命令。该命令可以显示相关接口的IPv6信息。在路由器Honeybee上使用该命令后的输出结果如示例3-8所示。

示例3-8 命令**show ipv6 interface**给出相关接口的IPv6信息，其中IPv6地址是EUI-64格式

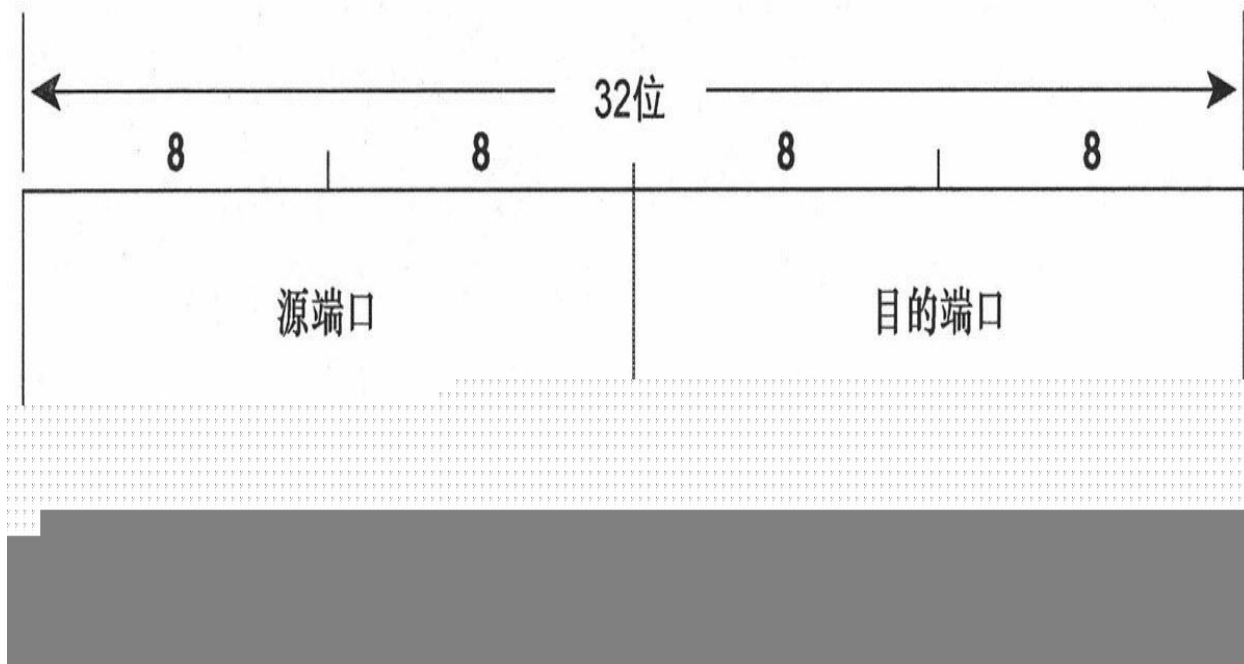


图3-3是一个使用IPv6地址的简单网络。 [\[5\]](#)

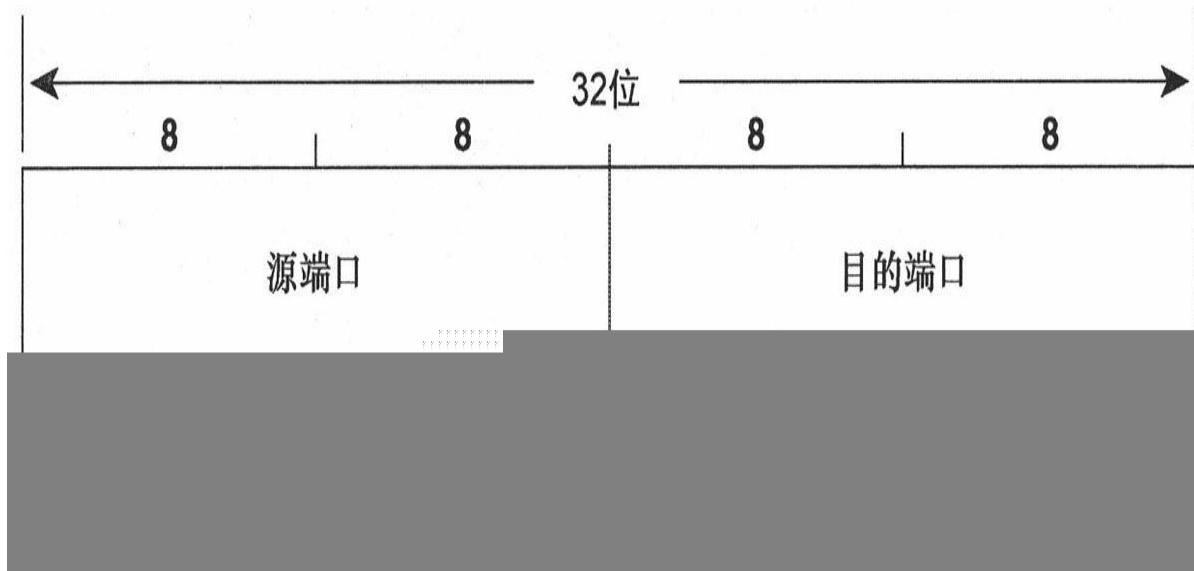
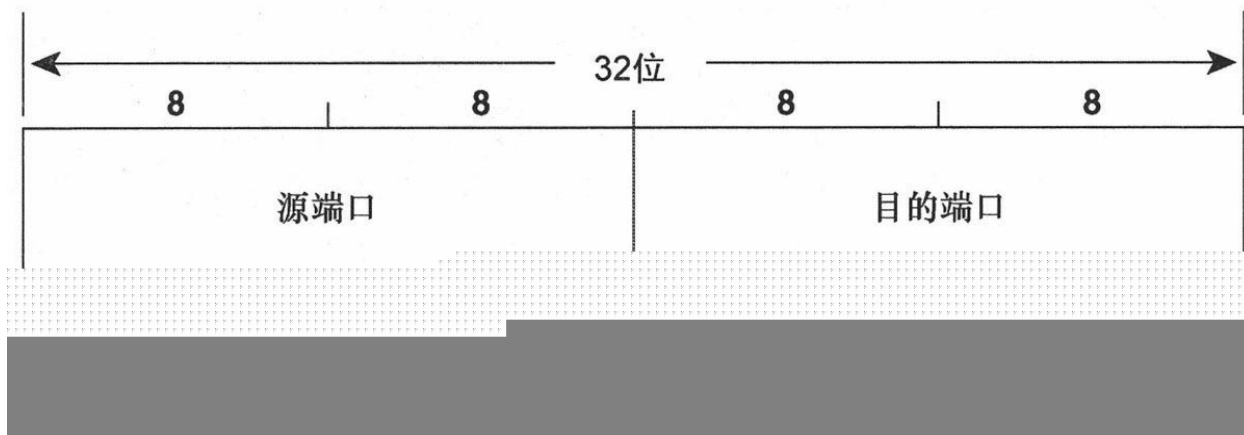


图3-3 静态路由选择使用IPv6也可以工作

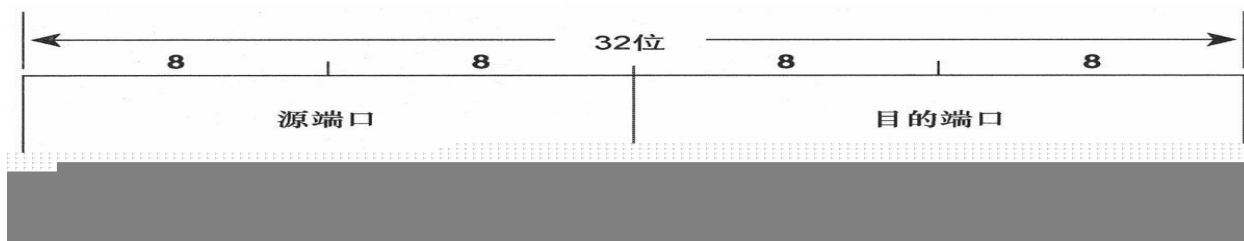
示例3-9给出了在Honeypot上配置静态IPv6路由的命令。

示例3-9 在路由器**Honeypot**上配置静态IPv6路由

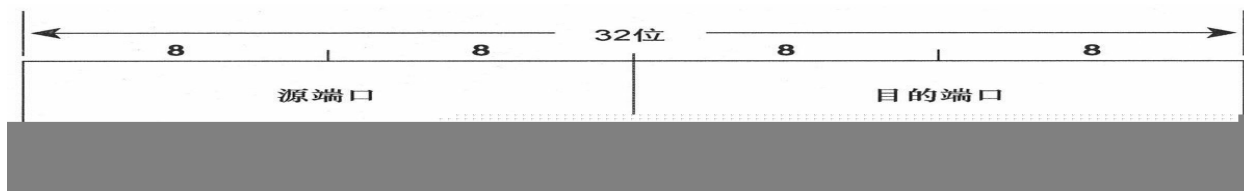


示例3-10和示例3-11分别给出了路由器Honeytree和Honeybee的路由表项。

示例3-10 为Honeytree配置IPv6静态路由



示例3-11 为Honeybee配置IPv6静态路由



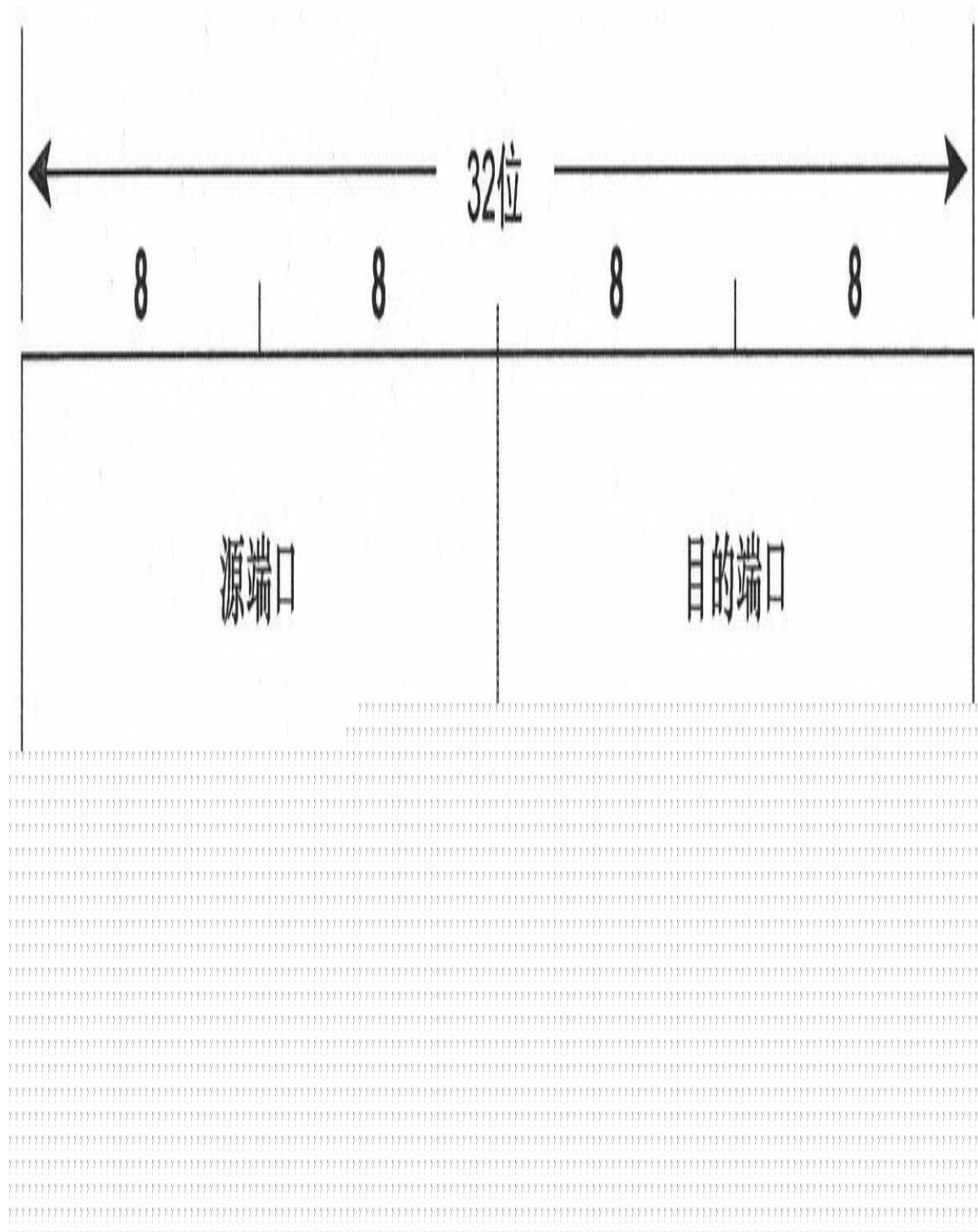
让我们看一下Honeypot和Honeytree的路由表项的下一跳地址。Honeypot每一条路由的下一跳地址是fec0::3:204:clff:fe50:flc0，Honeytree路由的下一跳地址是fec0::1:204:clff:fe50:flc0。这两个地址分别是Honeybee到Honeypot和Honeytree的接口地址。注意，Honeybee接口地址的后64位都是相同的；因为路由器使用第一个遇到的MAC地址来生成它的每个串行接口EUI-64格式的IPv6地址的后64位。

与IPv4一样，IPv6静态路由也可以使用出站接口代替下一跳地址。一种选择是在接口后面输入地址，你可以输入链路本地地址，或者一个配置

的地址。当出站接口是广播接口时，例如以太网，应该使用下一跳地址。

在示例3-12中，Honeypot在`ipv6 route`语句中仅指明了下一跳地址，因此得到示例的IPv6路由表。使用命令`show ipv6 route`可以显示出IPv6的路由表，其中包括前缀、前缀长度、下一跳地址或出站接口、管理距离和路由度量。

示例3-12 与IPv4一样，IPv6静态路由表包括目标网络和去往目标网络的下一跳地址

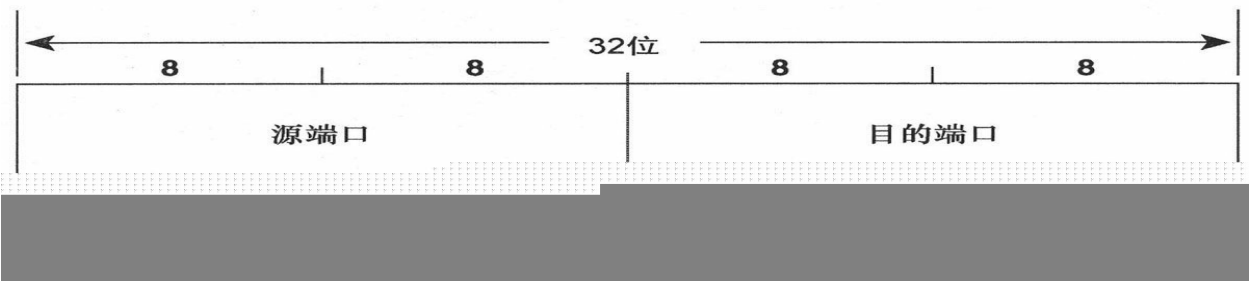


在示例3-12中，所示静态路由是使用下一跳IPv6地址建立的。同IPv4一

样，路由器必须递归查找与该IPv6地址相关联的出站接口。目标网络FEC0:0:0:A::/64表项的下一跳地址是FEC0::3:204:C1FF:FE50:F1C0，如果再进一步观察路由表，可以看到网络FEC0:0:0:3::/64连接到接口Serial0/0.2上。注意，使用下一跳IPv6地址建立的静态路由的管理距离是1，路由测度是0，这和IPv4是相同的。

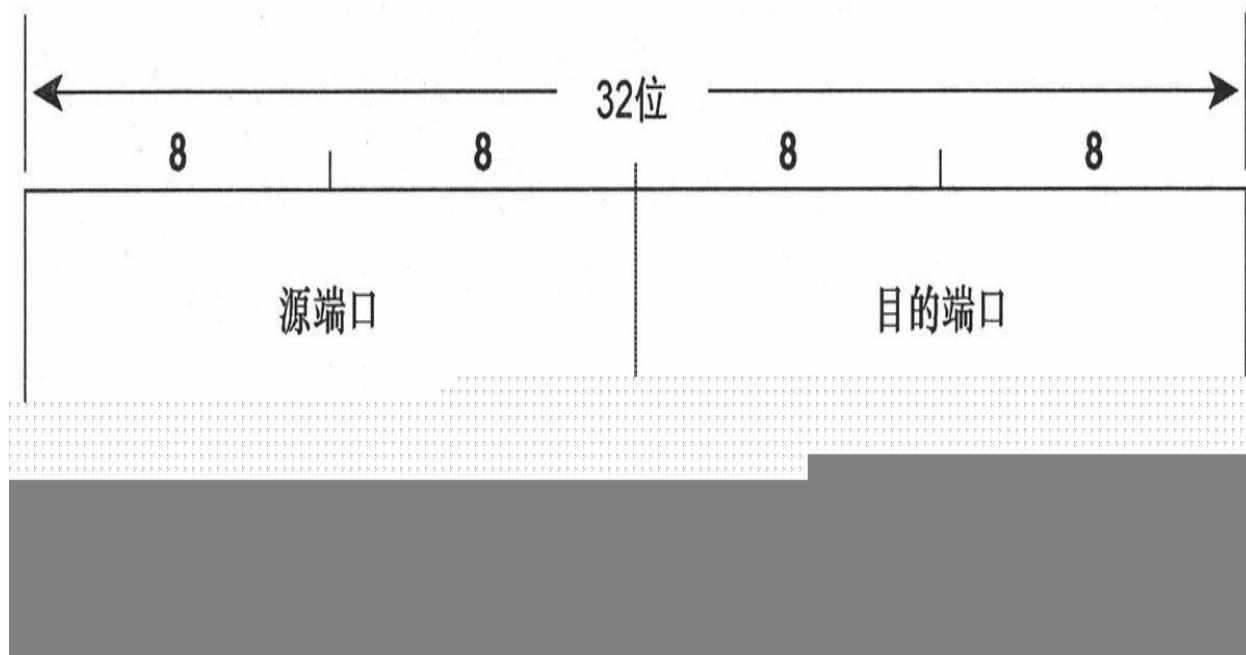
当然，我们还可以使用去往目标网络的出站接口或再结合下一跳地址来建立静态路由。如示例3-13所示，按照此方法对Honeypot的静态路由配置进行了修改。

示例3-13 另一种配置Honeypot静态路由的方法



其中最后一项使用了出站接口和下一跳地址，这样有助于阐明两种命令形式的区别。示例3-14显示了以此方法新生成的路由表。

示例3-14 在Honeypot上，使用出站接口替代下一跳地址后生成的路由表



另一个需要注意的问题是，虽然使用出站接口后静态路由的管理距离仍然为1，但是这和使用相同方法配置的IPv4路由却不相同，因为路由显示目标网络不是直接连接的。

除非同时指定出站接口和下一跳地址，否则在输入出站接口时下一跳地址是不确定的。在示例3-14的路由表中，语句1说目标网络通过::可达，出站接口是Serial0/0.2。“::”意味着下一跳没有被指定，但出站接口是Serial0/0.2。在点到点的串行接口上，不指定下一跳地址不会出现问题，因为在点到点网络的另一端仅会有一台设备，所有从出站接口发出的数据包一定能到达那台设备。

而在广播网络接口上，路由器必须找到邻居才可以发送数据包。路由器会在以太网上组播邻居请求消息，并等待下一跳设备的邻居通告。不同于移动IPv6节点，这里没有代理地址解析机制。因而对于以太网上具有去往目标网络路由的路由器来说，它不会代表其他设备响应邻居请求消息。

因此，在广播网络上使用出站接口配置静态路由时，必须指定下一跳地址。这里建议采用本地链路地址作为下一跳地址。这样做有两个原因，一是除非网卡或路由器被替换，否则本地链路地址不会发生改变，即使使用不同的IPv6全局前缀重新编号站点也是这样。二是可以与路由器通告信息中的地址保持一致，使用这些地址的进程可以按要求运行。例如

ICMPv6重定向进程。

路由器向广播网络上的所有设备通告自己的存在及本地链路地址。主机使用这些通告信息建立路由器列表，并使用这些列表确定如何向网络转发数据包。如果主机把一个数据包转发给路由器，并且该路由器知道一个更适合转发该数据包的路由器，那么路由器将向主机发送重定向信息。重定向信息包括另一台路由器的本地链路IPv6地址。当主机在处理重定向信息时，如果路由器列表中包含这台更合适的路由器，那么主机将向它转发数据包，否则（或列表中这台路由器使用了不同的IPv6地址），主机将丢弃数据包。

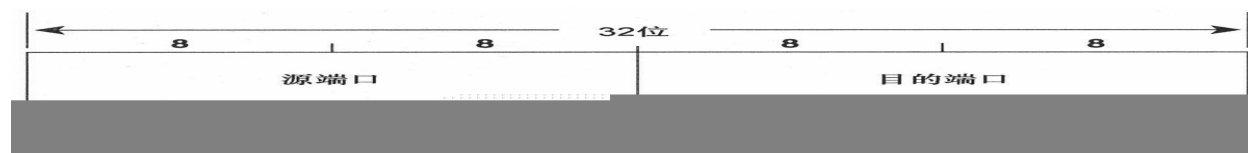
3.2.3 案例研究：汇总路由

汇总路由（**Summary Route**）是一个包含路由表中几个更加精确地址的地址。正是由于路由表项与地址掩码联合使用，使得静态路由的使用如此灵活。通过使用合适的子网掩码，有时可以为多个目标地址生成一条单一的汇总路由。

例如，在前面的案例研究中，为每个数据链路都使用了一条单独的路由。每个路由表项的掩码与连接数据链路的设备接口的地址掩码是一致的。再回想一下图3-2，读者可以发现对于路由器Piglet来说，可以使用经路由器Tigger可达10.4.0.0/16的单一表项来完成对子网10.4.6.0/24和10.4.7.0/24的说明。同样的，也可以用指向192.168.1.0/24的单一表项替代路由表中的子网192.168.1.0/27和192.168.1.64/27。10.4.0.0/16和192.16.1.0/24两个路由表项就是汇总路由。

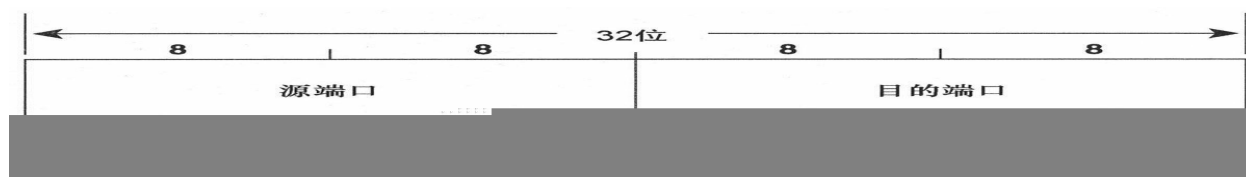
使用汇总路由技术，路由器Piglet的静态路由配置如示例3-15所示。

示例3-15 Piglet的静态路由被汇总为两条路由



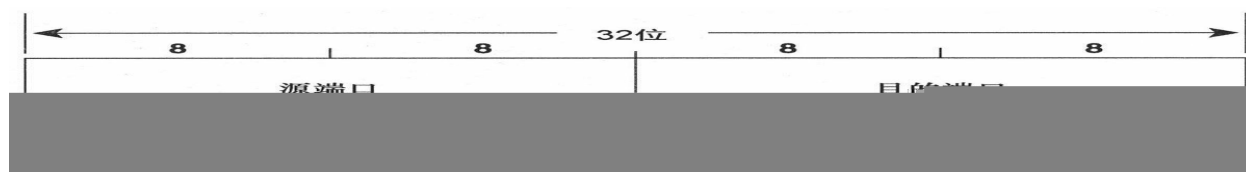
因为从路由器Pooh看，10.0.0.0网络的所有子网经过路由器Tigger都可以到达，所以仅需要包括主网地址和相应掩码的单一路由表项就够了（参见示例3-16）。

示例3-16 Pooh把所有关于10.0.0.0子网的路由汇总为一条路由



对路由器Eeyore，所有以192开头的目标地址都可以经过Tigger到达。如示例3-17所示，这个汇总路由中的地址掩码可以不指明所有的C类地址位。[\[6\]](#)

示例3-17 Eeyore把所有以192开头的路由都汇总为一条路由



汇总路由还可以应用到图3-3中所示的IPv6目标地址。通过把前缀长度由64改为62，可以把Honeypot的两条静态路由汇总成一条路由，这条路由包括从fec0:0:0:8::到fec0:0:0:b::的一组目标地址（参见示例3-18）。

示例3-18 Honeypot汇总IPv6静态路由



通过对一组子网甚至主网汇总，可以使静态路由项的数目迅速减小——在本示例中减少了三分之一。但是，在对地址进行汇总时需要小心，当汇总不正确时，可能会有意想不到的路由行为发生（见本章后面的3.3.1小节案例研究的内容）。第7章和第8章将会深入分析路由汇总及由此带来的问题。

[3.2.4 案例研究：选择路由](#)

如图3-4所示，在Pooh和Eeyore之间新增加了一条链路。除了去往主机10.4.7.25的数据包外，所有从Pooh到网络10.0.0.0的数据包都将使用这条新的路径。下面通过一条策略可以使这些数据包改经Tigger去往目的

地。在Pooh上，相应的静态路由命令见示例3-19。

示例3-19 Pooh上的静态路由命令可以实现一条策略，使流量经过指定的路由器

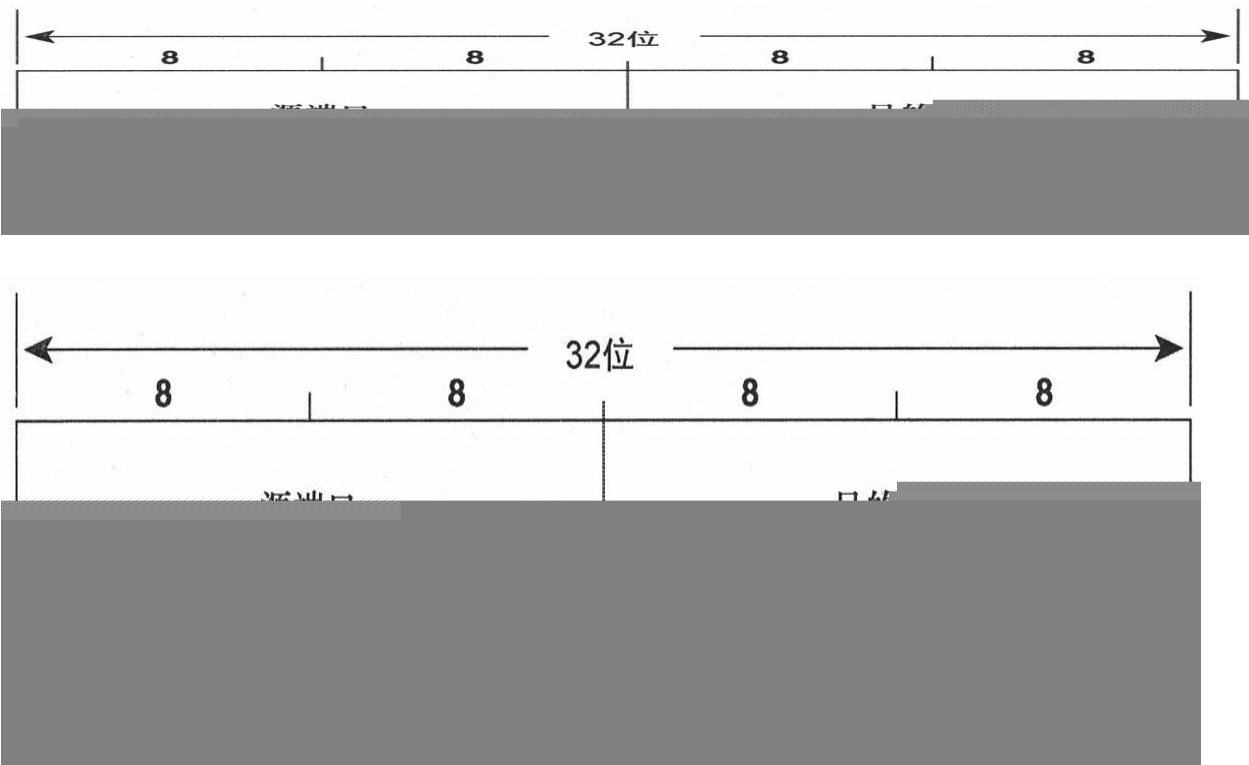
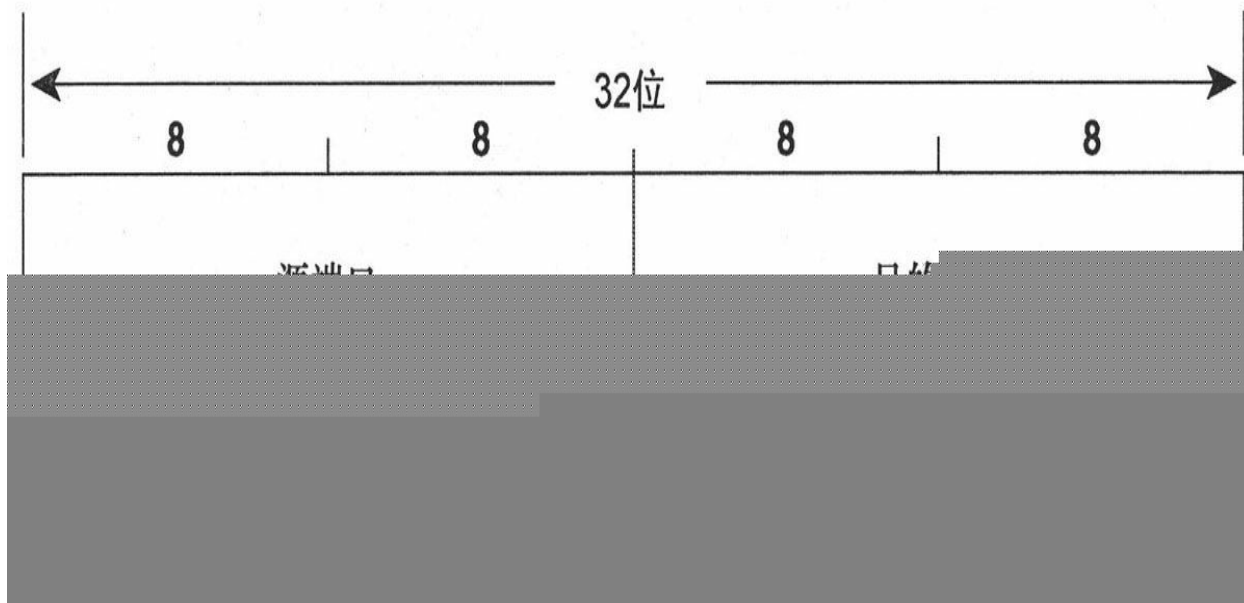


图3-4 在网络中添加了一条从Pooh到子网10.4.0.0更加直接的路径

目前，除了第2条路由指向Eyeore上新的接口192.168.1.34外，前两个路由表项没有其他变化。第3条路由是一条指向单一主机10.4.7.25的主机路由，通过把地址掩码所有位都设置为1便可以实现。注意，不同于10.0.0.0子网的其他条目，这条主机路由指向Tigger的接口192.168.1.66。

为了观察添加新路由条目之后数据包采用的路径，可以在路由器Pooh上打开调试功能**debug ip packet**（参见示例3-20）。这时从主机192.168.1.15有一个数据包被发向主机10.4.7.25。前两条调试捕获信息显示从接口E0进入路由器的数据包被路由到出站接口S0后再送往下一跳路由器192.168.1.66（Tigger），按照要求，在接口S0接收到的回复数据包被路由到出站接口E0后再发往主机192.168.1.15。

示例3-20 调试信息证实了在Pooh上的新路由表项工作正常



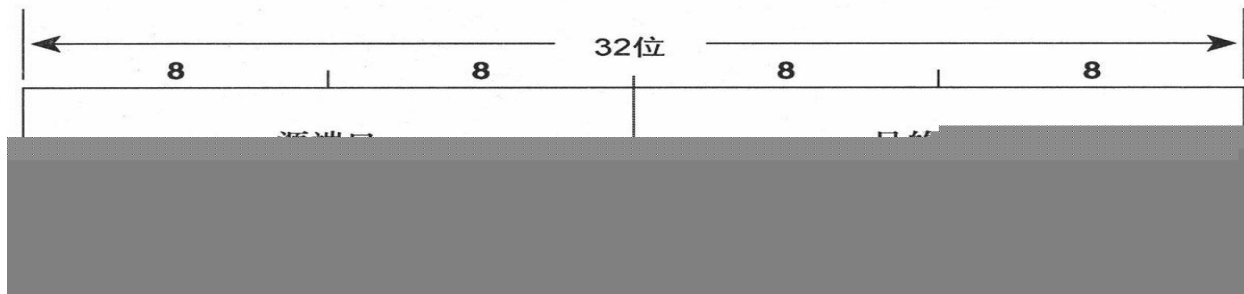
下一个调试信息显示一个数据包从主机192.168.1.15发往主机10.4.7.100。除了主机10.4.7.25外，去往10.0.0.0子网上所有主机的数据包都将沿新链路到达Eeyore的接口192.168.1.34。第3个调试信息证实了这一点。然而，第4个调试信息最初看上去有点让人惊讶。从10.4.7.100去往192.168.1.15的响应数据包自Tigger到达Pooh的接口S0。

还记得在其他路由器中的路由表项与原来例子中路由表项相比没有发生改变。不管是否期望有这样的结果，但是它体现出了静态路由的两个特性。

- 第一，如果网络拓扑结构发生变化，那么需要知道这些变化的路由器必须被重新配置。
- 第二，可以用静态路由建立非常精确的路由选择行为。在本例中，也许是来去流量使用的路径。

关于本例最后要说的一点是，由于使用了从Pooh到Eeyore再到Tigger的路径，而不是直接从Pooh到Tigger的路径，所以从Pooh到子网10.1.5.0的路由不是最优路径。示例3-21给出了一个更有效的配置。

示例3-21 在路由器**Pooh**配置更加有效的静态路由



现在，第3条路由将把去往子网10.1.5.0的所有数据包直接发送到Tigger。

3.2.5 案例研究：浮动静态路由（Floating Static Route）

浮动路由与其他静态路由不同，路由表中的其他路由总是优选于浮动静态路由，仅在一种特殊的情况下，即在一条首选路由发生失败的时候，浮动路由才会出现在路由表中。

在图3-5中，一台新的路由器（Rabbit）通过两条并行链路连接到Piglet上，一条链路连接它们各自的接口S0，另一跳连接各自的接口S1。添加第二条链路主要是考虑到线路冗余：如果主链路10.1.10.0发生故障，浮动静态路由将会指引流量经过备份链路10.1.20.0。

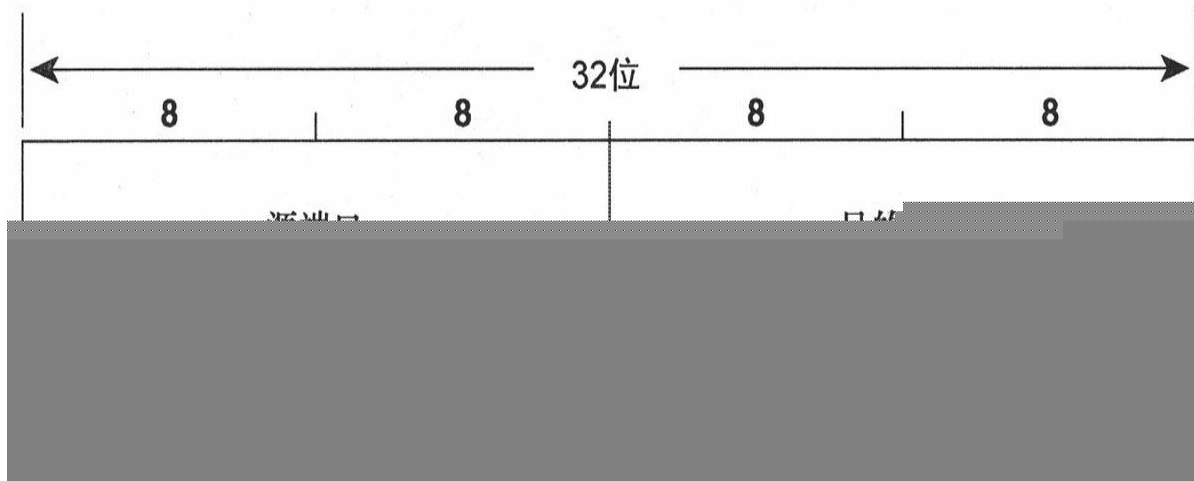


图3-5 Piglet上连接了一台新的路由器。这里使用了两条串行链路——一条作为主链路，另一条作为备份链路

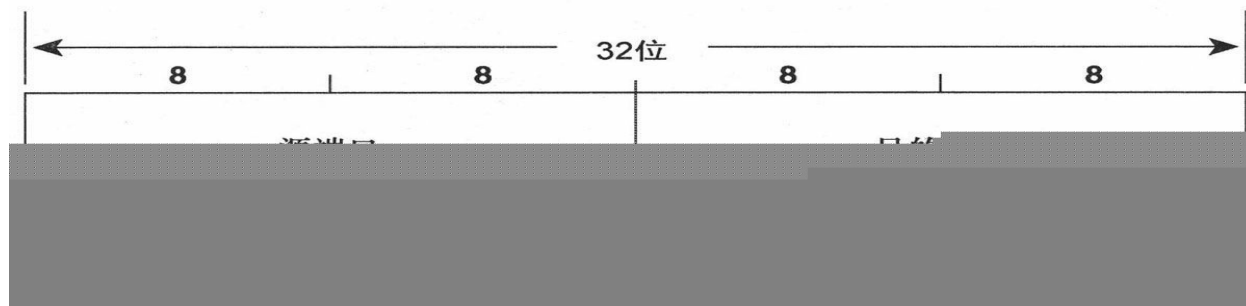
此外，Piglet以太网接口上的掩码由10.1.5.1/16改为10.1.5.1/24。这一改

变使得路由器Tigger上的一条路由不仅可以指向子网10.1.5.0，而且还可以指向新路由器的所有新子网。

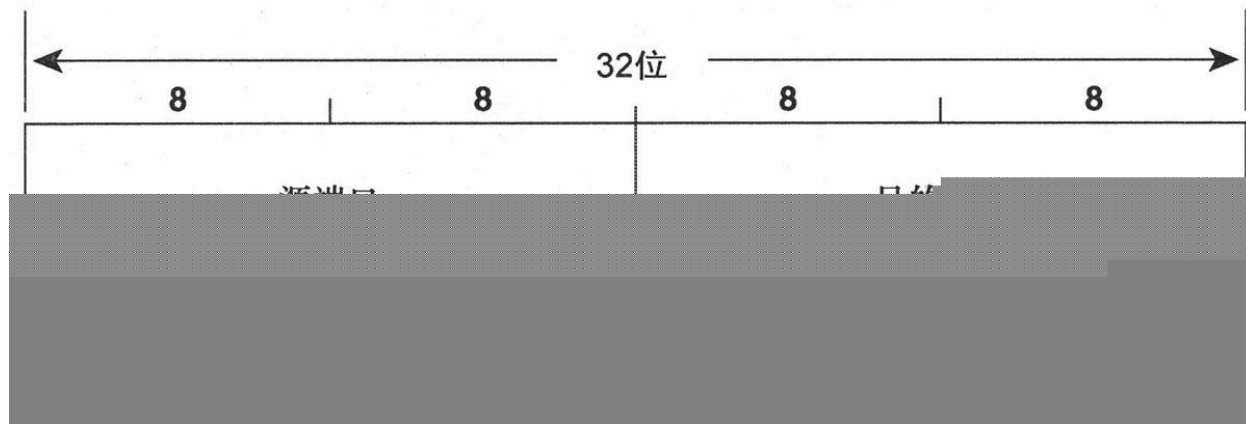
ip route 10.1.0.0 255.255.0.0 192.168.1.194

为了建立浮动静态路由，示例3-22和示例3-23分别给出了在Piglet和Rabbit上的路由表项。

示例3-22 为Piglet建立浮动静态路由的路由表项



示例3-23 为Rabbit建立浮动静态路由的路由表项



在Piglet上有两条路由指向Rabbit的网络10.1.30.0；一条指定Rabbit接口S0的地址作为下一跳地址，另一条指定Rabbit接口S1的地址作为下一跳地址。对每个目标网络Rabbit都有类似的双重表项。

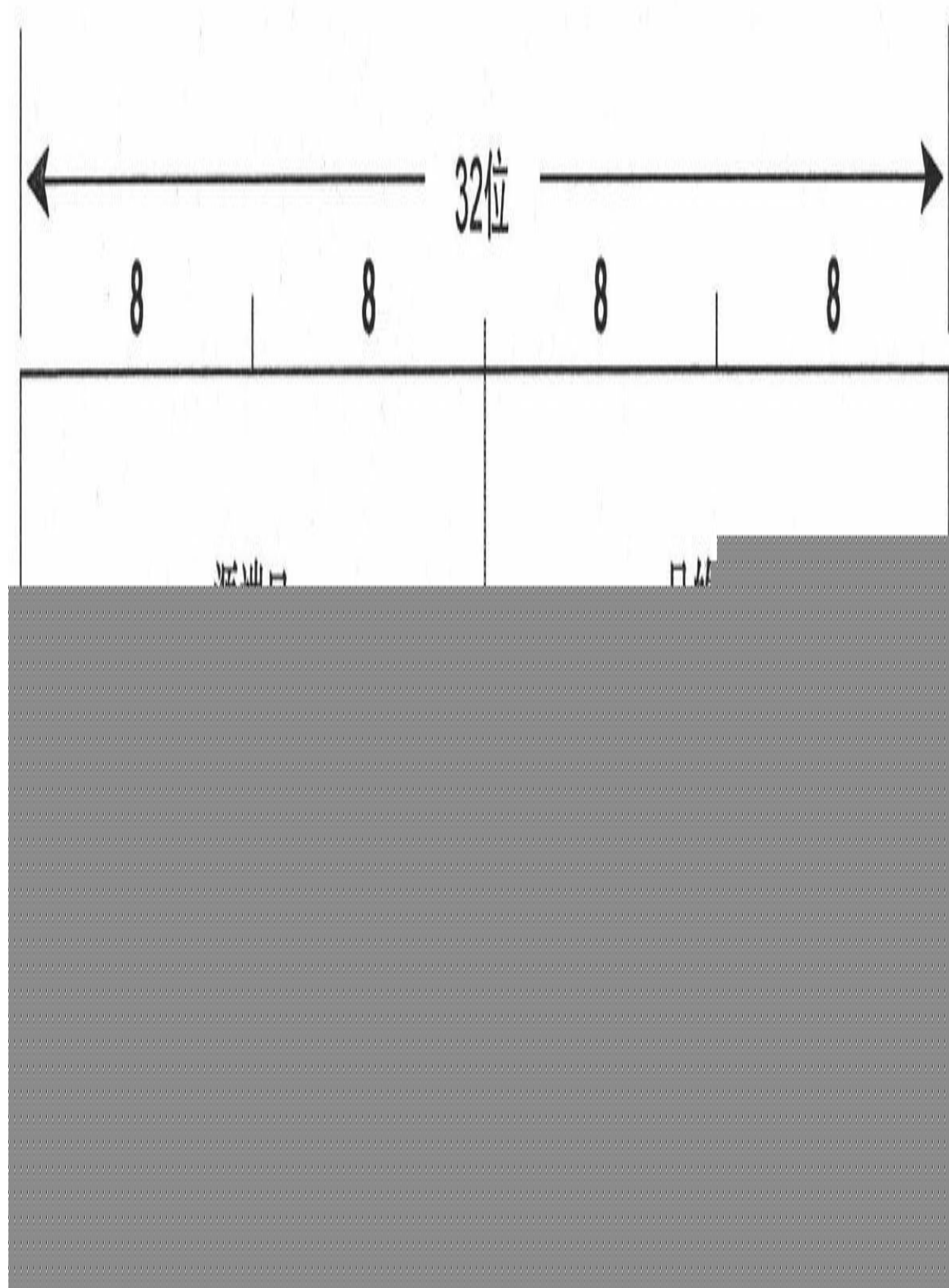
注意，在所有使用子网10.1.20.0的静态路由后面都跟有50。该数字指定

了管理距离，管理距离是一种优选度量。当存在两条路径到达相同的网络时，路由器将会选择管理距离较低的路径。这种思想初听起来有点像度量，但度量指明了路径的优先权，而管理距离指明了发现路由方式的优先权。

例如，指向下一跳地址的IPv4静态路由的管理距离为1，而指向出站接口的静态路由的管理距离为0。如果有两条静态路由指向相同的目标网络，一条指向下一跳地址，一条指向出站接口，那么后一条路由——管理距离值较低的路由——被选择。

将经过子网10.1.20.0的静态路由的管理距离提高到50，可以使经过子网10.1.10.0的静态路由成为首选路由。示例3-24反复3次给出了Rabbit的路由表。在第1个路由表中，所有指向非直连网络的路由都使用下一跳地址10.1.10.1。在每条路由表项中，括号内的数字指定了管理距离为1，度量值为0（因为静态路由没有度量）。

示例3-24 当主链路**10.1.10.0**失败时，备份链路**10.1.20.0**被启用。当主链路恢复时，再次启用主链路



接着，陷阱消息（trap message）通知连接到接口S0的主链路状态变为“down”，表明链路发生故障。查看第2次重复给出的路由表迭代，可以发现所有非直连网络路由都指向下一跳地址10.1.20.1。由于原来的首选路由不再可用，所以路由器切换到管理距离为50的备份链路。而且因为子网10.1.10.0发生故障，所以路由表中不再把它作为直连网络。

在第3次给出路由表之前，陷阱消息提示主链路状态恢复为“链路正常（up）”，路由表中再次显示子网10.1.10.0，而且路由器也再次使用10.1.10.1作为下一跳地址。

第11章将结合多种动态路由选择协议讨论管理距离，可以说动态路由选择协议的管理距离远远大于1。因此对于相同的目标网络，缺省情况下，到相同目标网络的静态路由总是优于动态路由。

3.2.6 案例研究：IPv6浮动静态路由

IPv6浮动静态路由的工作方式和IPv4一样。在图3-3中，如果Honeypot和Honeybee之间的链路发生故障，那么第二条链路将被添加到IPv6网络中（参见图3-6）。

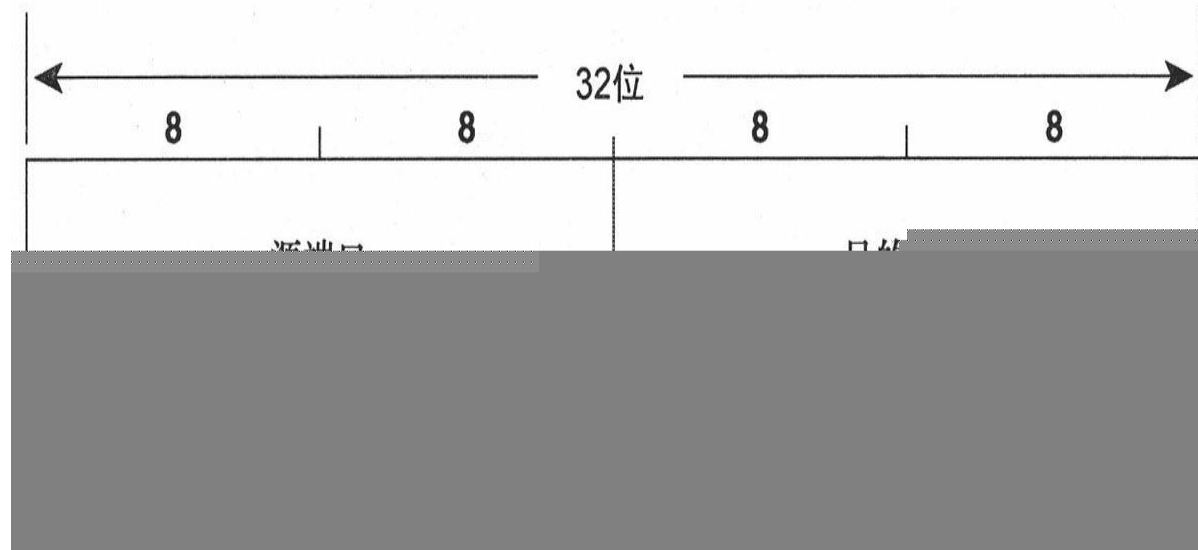
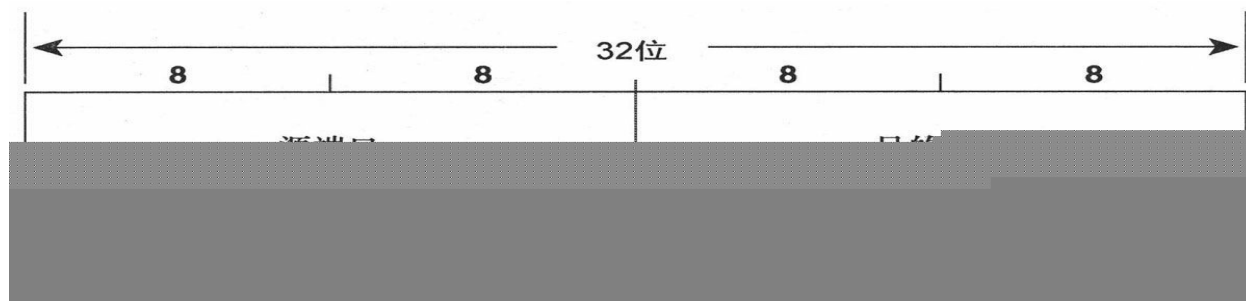


图3-6 使用浮动静态路由，路由器之间的备份链路可以在主链路中断时恢复网络畅通

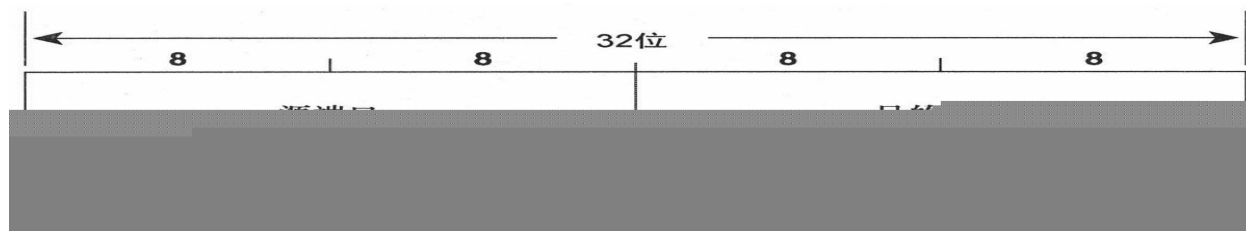
如示例3-25所示，新的静态路由表项被添加到Honeypot的配置中，其中

路由的管理距离大于1。在示例3-26中，类似的静态路由也被添加到Honeybee上。

示例3-25 为Honeypot和Honeybee之间新的冗余并行链路配置浮动静态路由



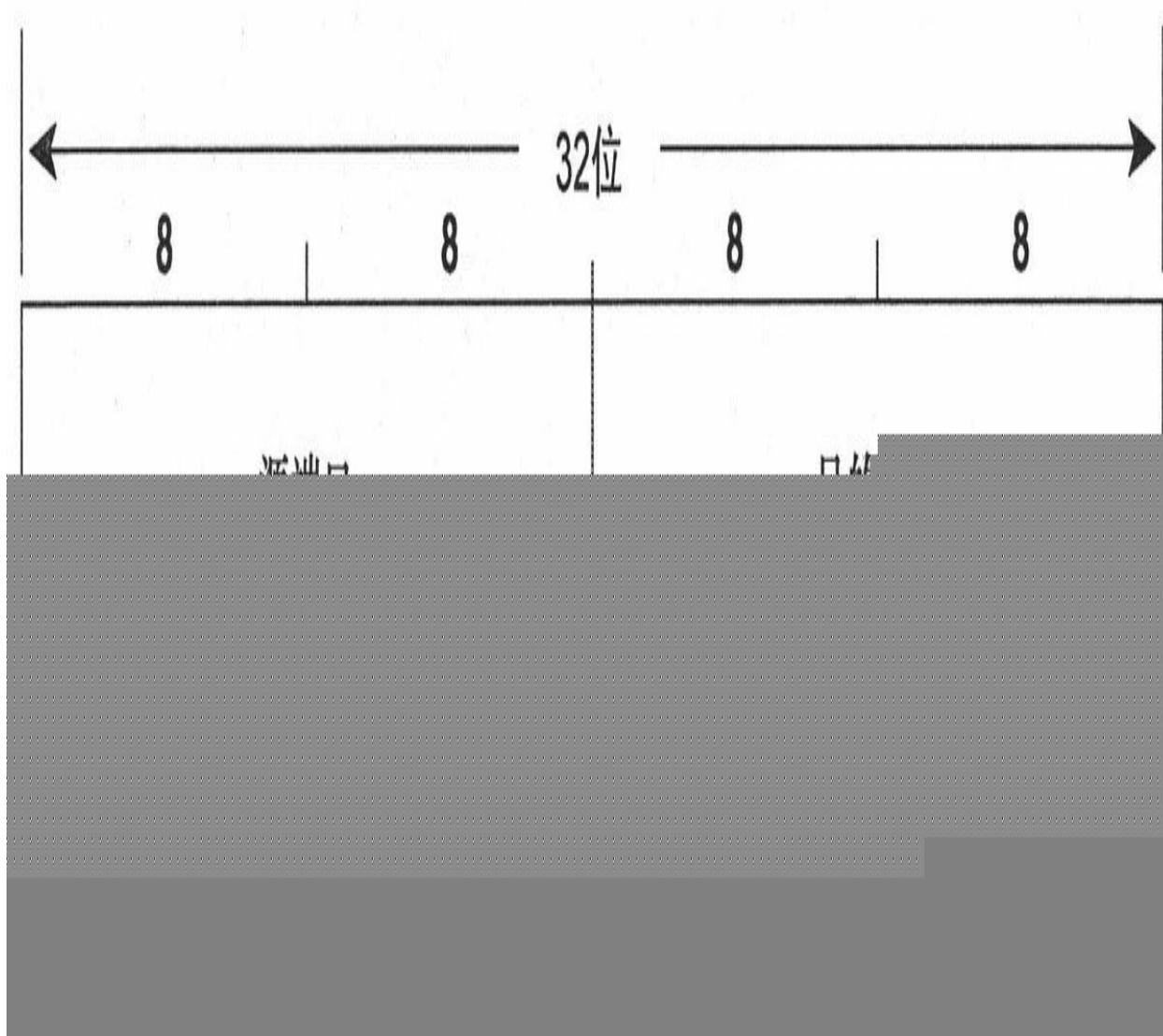
示例3-26 为Honeybee和Honeypot之间新的冗余并行链路配置浮动静态路由



除非链路发生故障，否则IPv6流量将从Honeypot的Serial0/0.2接口出站。

示例3-27给出了通过子网fec0::3:0:0:0/64安装到Honeypot路由表中的两条路由，这两条路由的管理距离都为1。当接口S0/0.2发生故障时，管理距离为50的备份路由替代了原来的路由；如果接口S0/0.2恢复正常后，路由选择进程将会学习去往目标网络的更优路由，并重新将其安装到路由表中。

示例3-27 仅当管理距离小的路由被删除时，管理距离大的静态路由才会被添加到IPv6的路由表中



不管是使用出站接口还是下一跳地址，IPv6静态路由管理距离的缺省值都为1。使用这两种方式指定的静态路由是等价的，并且会均分负载（在3.2.7小节中学习）。

[3.2.7 案例研究：均分负载](#)

上一小节所用配置方法的弊病是在正常情况下备份链路不能被利用；备份链路的可用带宽资源被浪费了。均分负载（Load Sharing），又叫负载均衡，允许路由器利用多路径的优点，在所有可用的路径上发送数据包。

均分负载可以是等价或非等价的，这里的代价（cost）是一个通用术

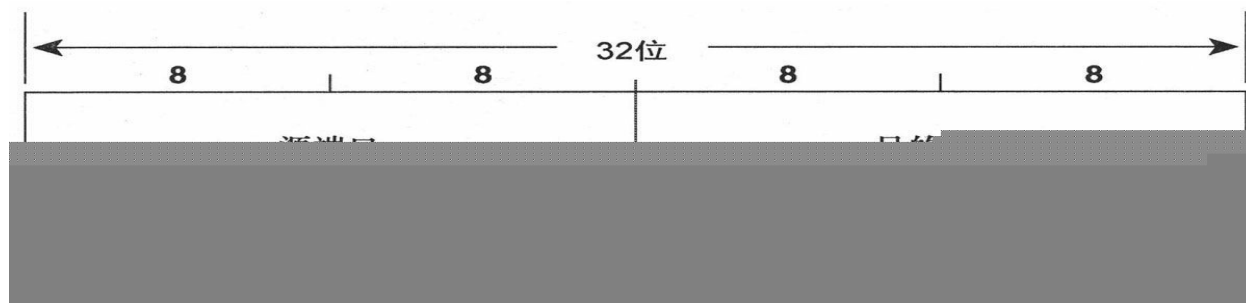
语，指与路由相关联的度量。

- 等价均分负载（**Equal-Cost Load Sharing**）——将流量均匀地分布到多条度量相同的路径上。在这种情况下，均分负载又叫负载平衡。
- 非等价均分负载（**Unequal-Cost Load Sharing**）——将数据包分布到度量不同的多条路径上。各条路径上分布的流量与路由代价成反比。也就是说，代价越低的路径分配的流量越多，代价越高的路径分配的流量越少。

一些路由选择协议可以支持等价和非等价负载均衡两种方式，而其他一些路由选择协议仅支持等价方式。静态路由没有度量，所以仅支持等价负载均衡。

为了使用静态路由实现负载均衡，在图3-5中配置了并行链路，示例3-28和示例3-29分别给出了Piglet和Rabbit相应的路由表项。

示例3-28 为实现均分负载使用静态路由配置并行链路： **Piglet**上相应的路由表项

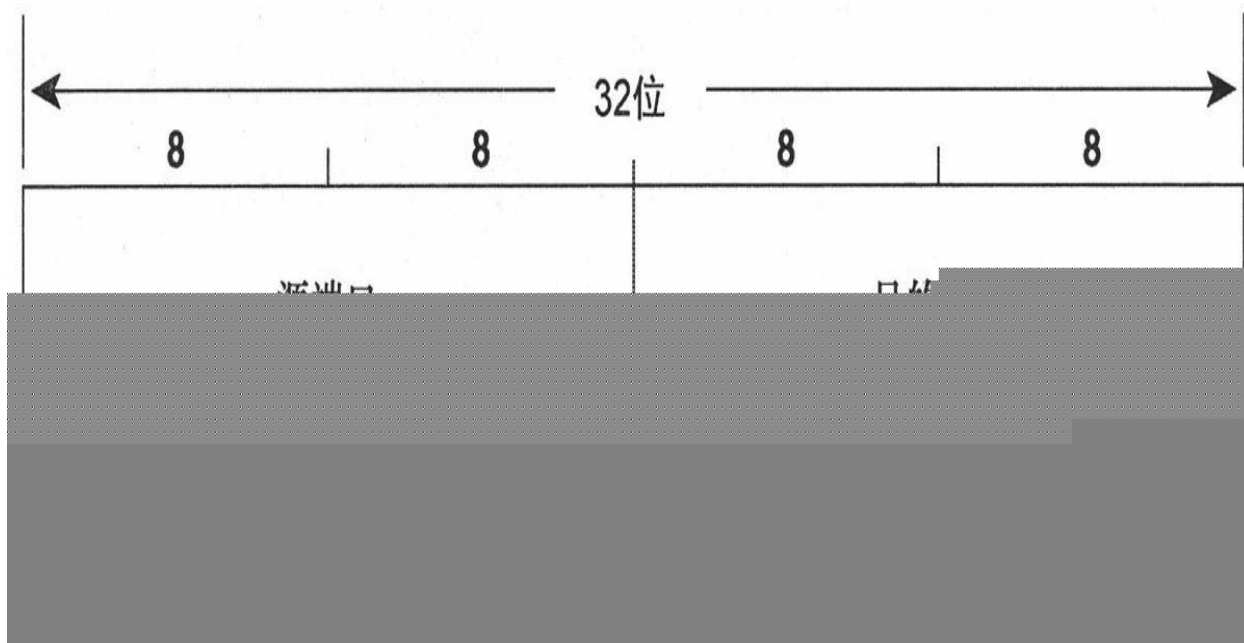


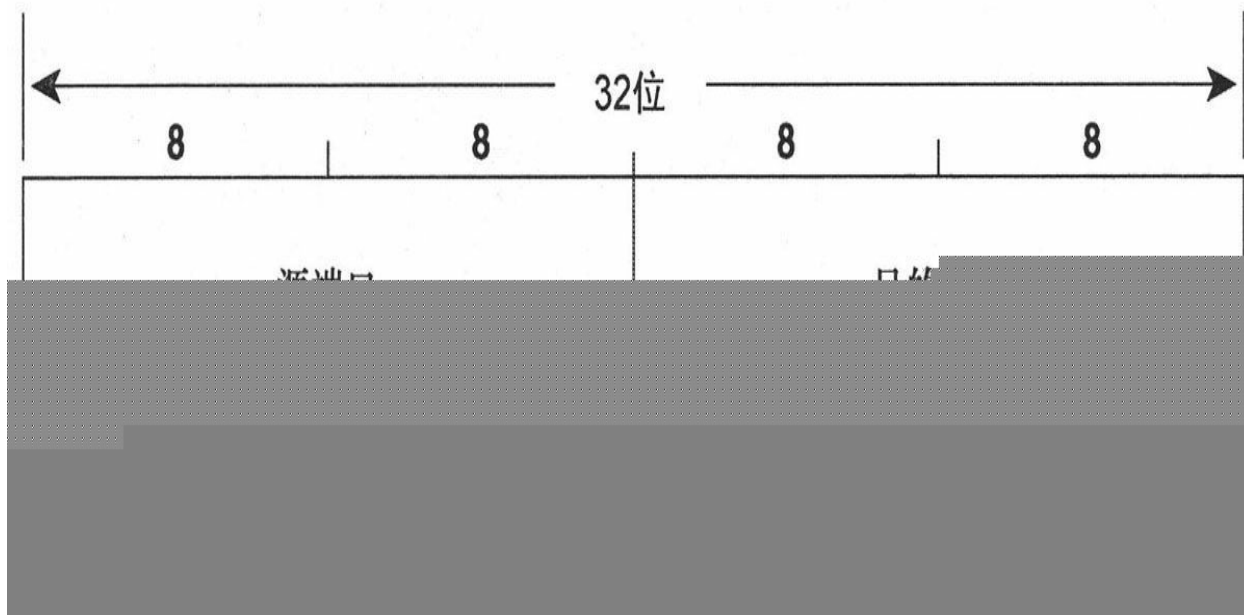
示例3-29 为实现均分负载使用静态路由配置并行链路： **Rabbit**上相应的路由表项



除了两条链路使用缺省管理距离1以外，这些路由表项已在上一节的浮动静态路由中被用过了。如示例3-30所示，在Rabbit的路由表中，对于每个目标网络都存在两条路由。

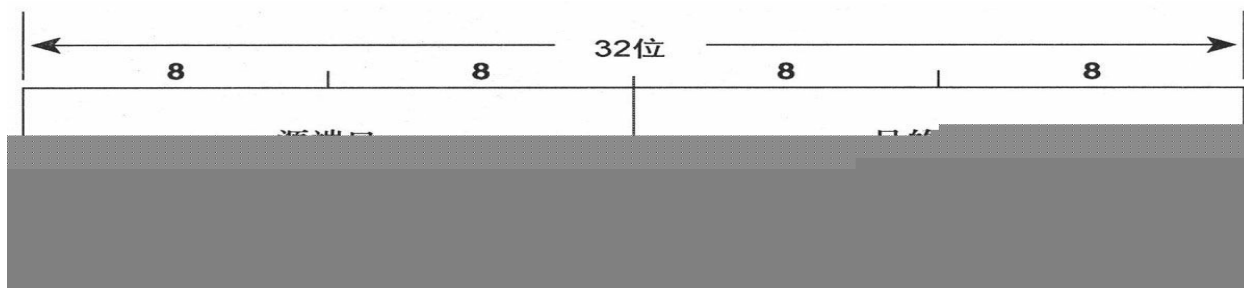
示例3-30 这个路由表显示出对于每个目标网络，都存在两条路径。路由器将会在多条路径上平衡负载



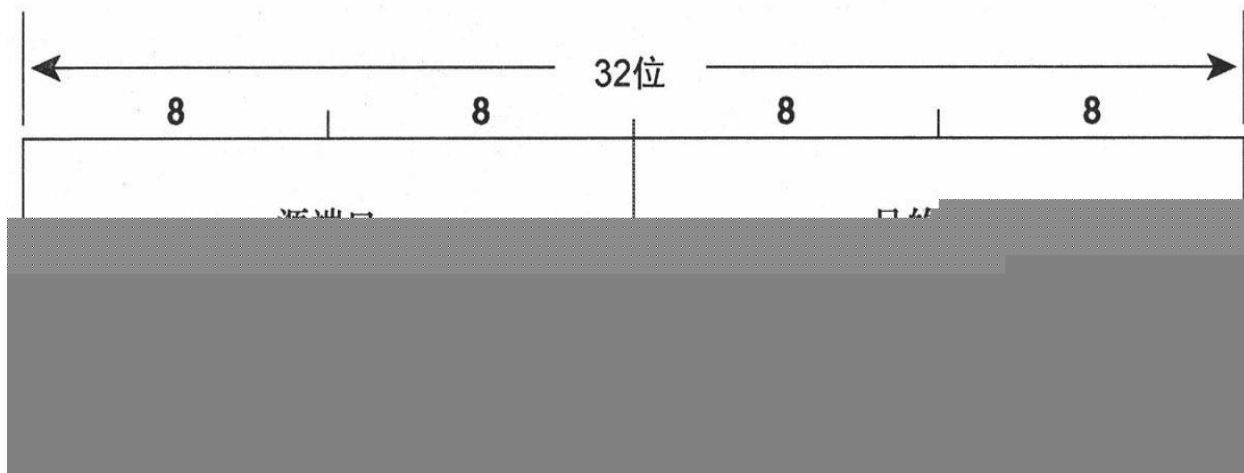


IPv6和IPv4工作方式一样。在示例3-31中，Honeypot的路由配置将会产生示例3-32所示的路由表。

示例3-31 为了使Honeypot的IPv6静态路由可以实现均分负载，对每个地址前缀使用不同的下一跳地址



示例3-32 路由器在去往相同目标网络的两条路径上均分负载



负载均衡有两种方式：基于目标网络和基于数据包。

1. 负载均衡和Cisco急速转发

基于目标网络的负载均衡是根据目标地址分配负载。假设到一个网络存在两条路径，那么发往该网络中第一个目标的数据包从第一条路径通过，发往网络中第二个目标的数据包通过第二条路径，发往此网络中第三个目标的所有数据包还通过第一条路径，依此类推。这就是Cisco急速转发（CEF）使用的缺省负载均衡方式。在大部分平台上，IPv4的缺省交换模式是CEF，但是IPv6却不是。

CEF是一种非常有效的交换方法。它事先从路由表中获取信息并把信息存储在转发信息库（FIB）中，当任何数据包需要这些信息时可以立即使用。FIB包括路由表中的所有目标网络，如果路由表稳定且不发生改变，那么FIB也不会变化。CEF使用一个单独的数据表—邻接关系表，为FIB的每个表项维护第二层转发信息。邻接关系表由第二层信息构成，例如，这些信息可以通过IPARP或IPv6邻居发现协议学习到。FIB和邻接关系表是在数据包转发之前建立的。

CEF在缺省情况下是基于目标进行负载均衡。这实际上是按照源目地址对进行负载均衡。所有发往特定目标地址的流量只要源地址相同都会从相同的接口出站，而不同源目地址对的流量可能会从下一个接口出站。

基于数据包的负载均衡是交换IPv4数据包的另一种方式。对于IPv6，CEF仅支持基于目标网络的负载均衡方式。基于数据包的负载均衡方式意味着在不同的链路上发送数据包，即使在路径等代价、目标相同的情

况下也是这样。如果路径代价不同，那么可能会在高、低代价路径上按照代价比率进行分流。基于数据包的负载均衡方式可以更加均匀地布流量，这取决于不同源目地址对的数量。但是数据包选择不同的路径去往目标网络会引起非顺序到达。对于某些应用来说，这是不能接受的，例如VoIP。

为了确定CEF功能是否在路由器上被全局的打开，可以使用命令**show ip cef** 和**show ipv6 cef**。如果缺省情况下CEF没有被打开，针对IPv4可以使用命令**ip cef**，而对IPv6来说，必须先打开IPv4的CEF，然后使用命令**ipv6 cef** 打开此功能。

在IPv4下，命令**ip load-sharing per-packet** 可以打开基于数据包的负载均衡功能，如果需要打开基于目标地址的负载均衡，可以用**ip load-sharing per-destination** 命令。你可以使用命令**show cef interface** 来检查使用了哪一种负载均衡模式，该命令可以给出在这个接口上配置的CEF信息。

路由器通常根据入站接口和源与目的地址类型确定是否使用CEF交换。对于考虑使用CEF的路由器来说，出站接口必须配置为CEF交换模式，如果接口上配置了CEF，那么CEF将尝试交换数据包。否则，CEF将会把数据包交付给仅次于最好的可用交换方法去处理。对于IPv4，这种方法是快速交换，而在IPv6中叫进程交换（process switching）。

你可以使用命令**show cef interface {interface}** 和**show ipv6 cef {interface} detail** 来验证在接口上CEF功能是否被打开。

2. 基于目标网络的负载均衡和快速交换

IOS在配置了快速交换的出站接口上执行基于目标网络的负载均衡，在一些路由器上，IOS的缺省交换模式是快速交换。

快速交换的工作方式如下：

（1）当路由器为第一个去往特定目标的数据包进行交换处理时，路由器将执行路由表查询并选择出站接口。

（2）然后获取有关被选接口的数据链路信息（例如从ARP表），这些信息对数据包成帧是必需的，最后封装数据包并发送。

(3) 前面获取的路由和数据链路信息被输入到快速交换的高速缓冲内。

(4) 一旦去往相同目的地的后继数据包进入路由器，高速缓冲中的信息使路由器不必查找路由表和ARP高速缓冲，就可以立即交换数据包。

快速交换意味着所有去往指定目的地的数据包都从相同的接口被发送出去，因此交换时间和处理器的占用率会大大降低。当去往相同网络内不同主机的数据包进入路由器且还存在一条可选路由时，路由器会在另一条路径上发送数据包到目的地。因此路由器能够做得最好的就是基于目标网络的均衡负载。

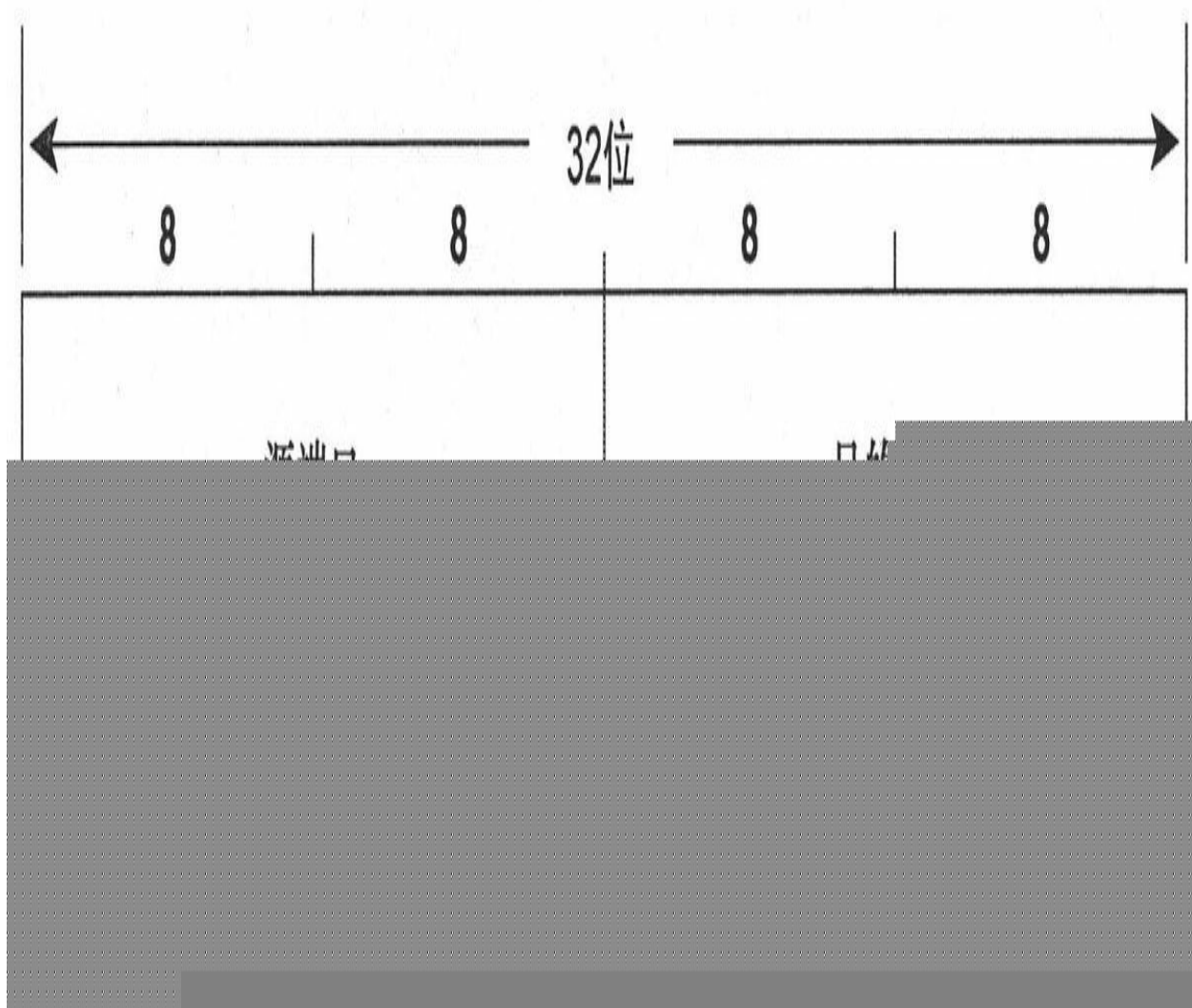
3. 基于数据包的负载均衡和过程交换

过程交换（**process switching**）就是对于每个数据包，路由器都要进行路由表查询和接口选择，然后再查询数据链路信息。因为每一次为数据包确定路由的过程都是相互独立的，所以不会强制去往相同目标网络的所有数据包使用相同的接口。为了在接口上打开过程交换功能，可以在IPv4下使用命令**no ip route-cache**。对于IPv6什么也不需要做，因为缺省情况下该功能是打开的。

在示例3-33中，主机192.168.1.15向主机10.1.30.25发送了6个ping。在Piglet上使用**debug ip packet**可以观察到ICMP的回应请求和回应应答数据包。通过查看出站接口和转发地址可以发现Piglet和Rabbit都在交替使用接口S0和S1。注意命令**debug ip packet**仅允许观察过程交换的数据包，快速交换的数据包将不被显示出来。

注意：命令**debug ip packet**仅显示过程交换的数据包。

示例3-33 路由器交替使用接口S0和S1向相同目标网络发送数据包。
注意，在两条链路另一端的路由器也以同样的方式回复数据包



正如许多设计选择一样，基于数据包的均分负载也是要付出代价的。这种方式虽然使流量的分布比前一种方式更均匀，但是快速交换的较低交换时间和处理器占用的优点也随之丧失了。

4. 哪一种交换方法会被用到

IOS首先基于入站接口的配置来决定交换模式；如果接口上配置了CEF，不管出站接口的配置是什么，数据包都会被CEF交换。

如果入站接口上没有配置CEF，那么IOS会处理并转发数据包，并根据出站接口的配置，后继的数据包或者被快速交换，或者被过程交换。表3-1给出了交换方法与出/入站接口配置的对应关系表。

表3-1 IOS根据入站和出站接口的配置确定交换方法

入站配置	出站配置	所用的交换方法
CEF	过程	CEF
CEF	快速	CEF
过程	CEF	快速（如果是IPv6为过程）
过程	快速	快速
快速	CEF	快速（如果是IPv6为过程）
快速	过程	过程

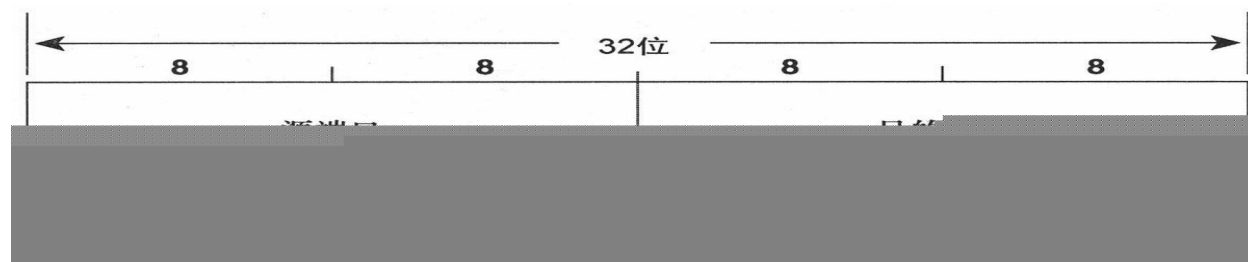
如果入站接口的CEF功能被打开，IOS将只使用CEF交换数据包。否则，出站接口的配置会确定交换方法。注意，如果在出站接口上打开CEF功能的同时又在入站接口上配置了过程交换或快速交换，那么快速交换将被使用。只有在入站接口上配置了CEF，它才会起作用。对于IPv4，尽管在出站接口上打开了CEF功能，但是起作用的还是快速交换。

有些时候即使打开了CEF，但是并没有使用CEF交换数据包（例如，如果访问列表的日志功能被打开和数据包将被记录下来）。那么数据包将被送交仅次于最快的交换方法，例如在IPv4下使用快速交换，在IPv6下使用过程交换。

3.2.8 案例研究：递归表查询

所有路由表项不必一定指向下一跳路由器。图3-7是图3-5中网络的简化版。在这个网络中，Pooh配置见示例3-34。

示例3-34 Pooh的路由表项使用了多种地址作为下一跳参数。这些地址不需要是下一跳路由器接口的实际地址



如果Pooh需要向主机10.1.30.25发送数据包，Pooh将查找路由表并发现经过10.1.10.2可以到达这个子网。因为这个地址不在直连网络中，所以Pooh必须再次查找路由表并发现去往10.1.10.0需要途经192.168.1.194。由于这个子网也不是直连子网，因而需要进行第3次路由表查找。这次Pooh发现途经192.168.1.66可以到达192.168.1.192，并且192.168.1.66在一个直连子网中。于是现在可以转发数据包了。



图3-7 为了到达网络10.1.30.0，Pooh必须进行3次路由表查找

因为每次路由表查询都会花费处理器的时间，所以在正常情况下强制路由器进行多次路由表查询是一种不好的设计决策。快速交换对递归查询进行了限制，仅对去往每个目标网络的第1个进行递归查询，从而有效地降低了这些不利的影响；但是在使用这种设计方法之前仍然需要充分的理由。

图3-8给出了一个递归路由表查询的实例，例子中的递归查询也许是有用的。在这里，Sanderz途经Heffalump可以到达所有网络。然而，网络管理员计划弃用Heffalump，改用Woozle。Sanderz中的前12条路由不再指向Heffalump，而是指向被连接到子网10.87.14.0上的合适路由器。最后一条路由指明经Heffalump可以到达子网10.87.14.0。

使用下面的配置，仅需要改动最后一条静态路由，便可以使Sanderz的所有路由都重新指向Woozel，详见示例3-35。

示例3-35 仅需改动最后一条静态路由，便可以使Sanderz的所有路由都重新指向Wooze l



如果以10.23.5.20作为所有静态路由条目的下一跳地址，那么需要删除

13条路由，再重新输入13条新的路由。不过，读者要仔细斟酌一下，省去重新输入静态路由的麻烦与递归查询带给路由器的额外负担到底谁更重要。

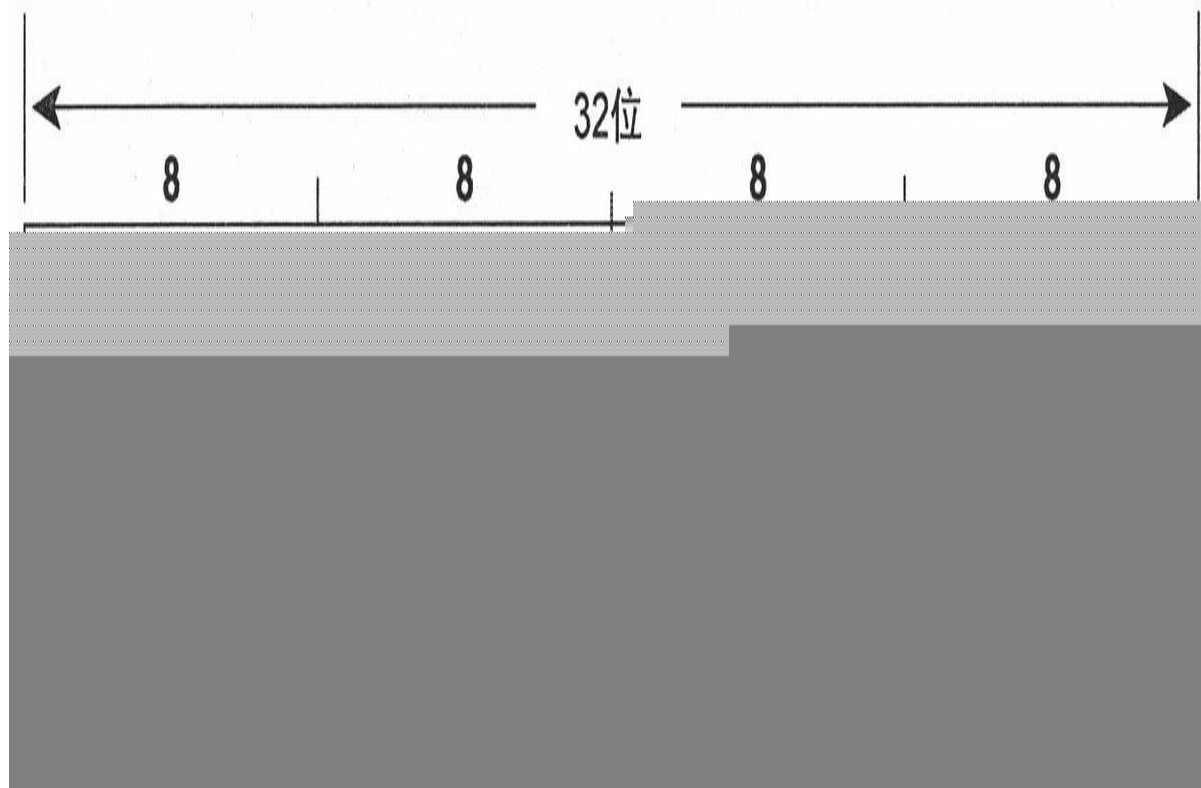


图3-8 为Sanderz配置递归查询，使网络管理员仅需要修改一条路由，便可把从Sanderz到Heffaoump的流量重定向到Woozle

3.3 静态路由故障诊断

也许你听过这样的问题，“他知道什么？他什么时候知道的？”这些问题对于网络调查员也同样适用。当排除路由故障时，首要的一步就是检查路由表。路由器知道什么？这些信息在路由表中存在多久了？路由器知道怎样如何到达讨论中的目标网络？路由表中的信息准确吗？为了顺利地排除网络的故障，了解如何跟踪路由是十分必要的。

3.3.1 案例研究：追踪故障路由

图3-9所示的网络在前面已经作过相应的配置，其中包括每台路由器相应的静态路由。现在发现一个问题，在连接到Pooh以太网接口的子网192.168.1.0/27上，设备与子网10.1.0.0/16上的设备可以正常地通信。然而，当从Pooh向子网10.1.0.0/16发送ping时，结果出现ping失败（参见示例3-36）。这看上去很奇怪。如果Pooh可以成功地路由数据包到达目标网络，那么为什么从Pooh发的数据包却传送失败呢？

为了解决这个问题，需要跟踪ping所经过的路径。首先，检查Pooh的路由表（参见示例3-37）。（根据路由表）目标地址10.1.5.1可以匹配到路由表项10.0.0.0/8，该子网经下一跳地址是192.168.1.34，192.168.1.34是Eeyore的一个接口。

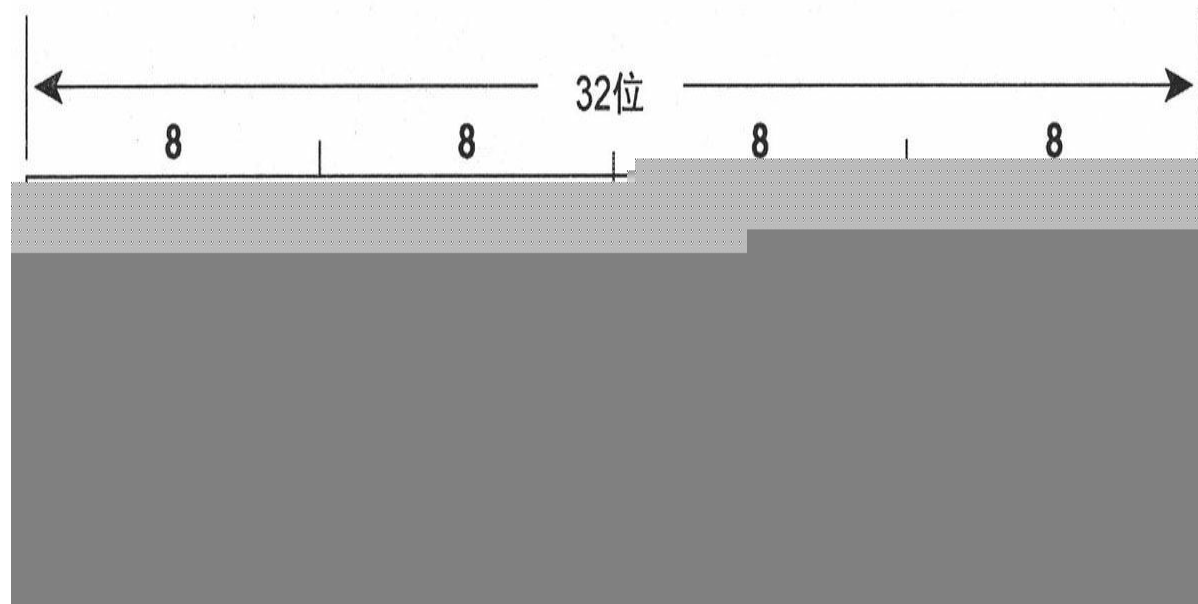
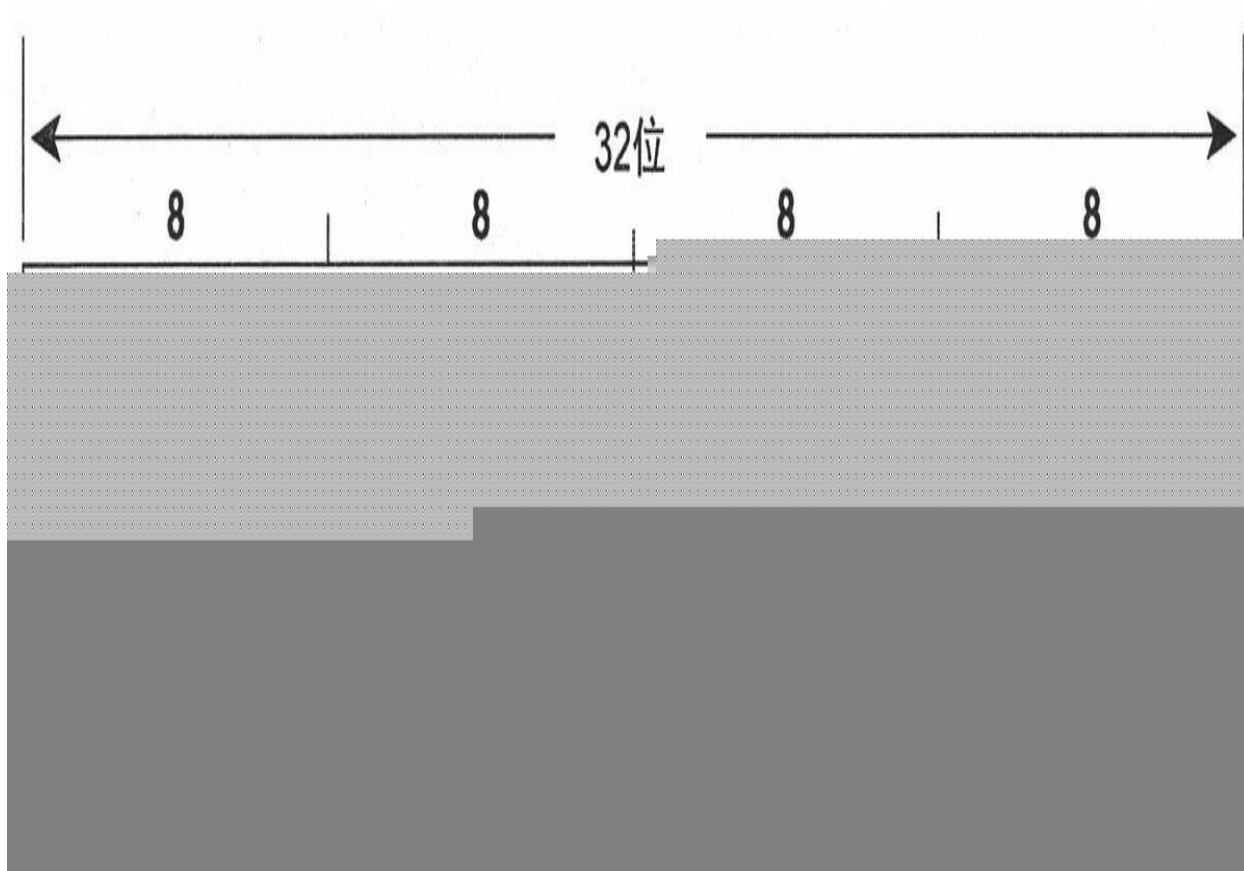
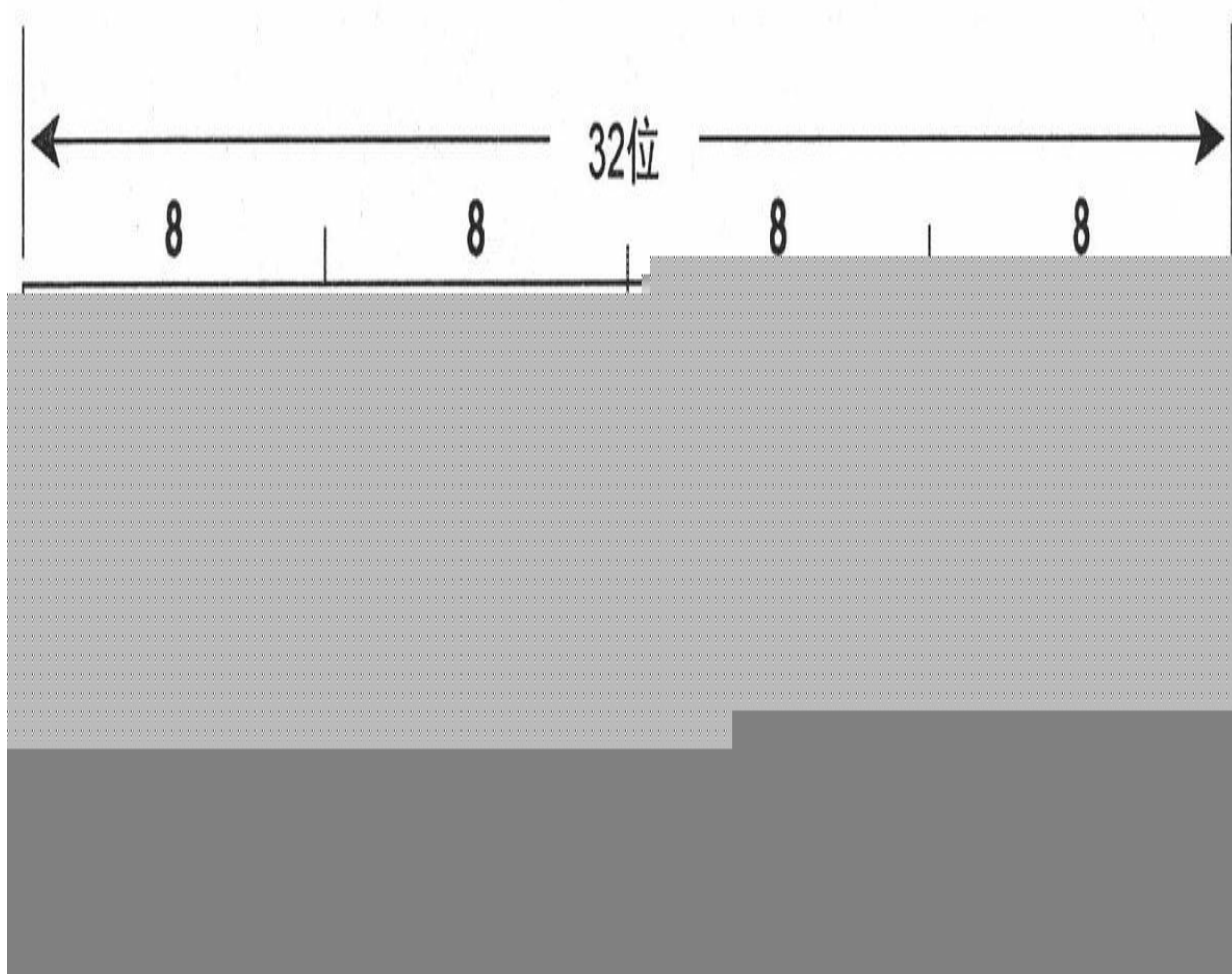


图3-9 从子网192.168.1.0/27到子网10.1.0.0/16数据包可以被正确地路由，但从Pooh却不能ping通子网10.1.0.0/16中的任何设备

示例3-36 子网192.168.1.0/27上的设备可以ping通Piglet的以太网接口，但Pooh却Ping不通

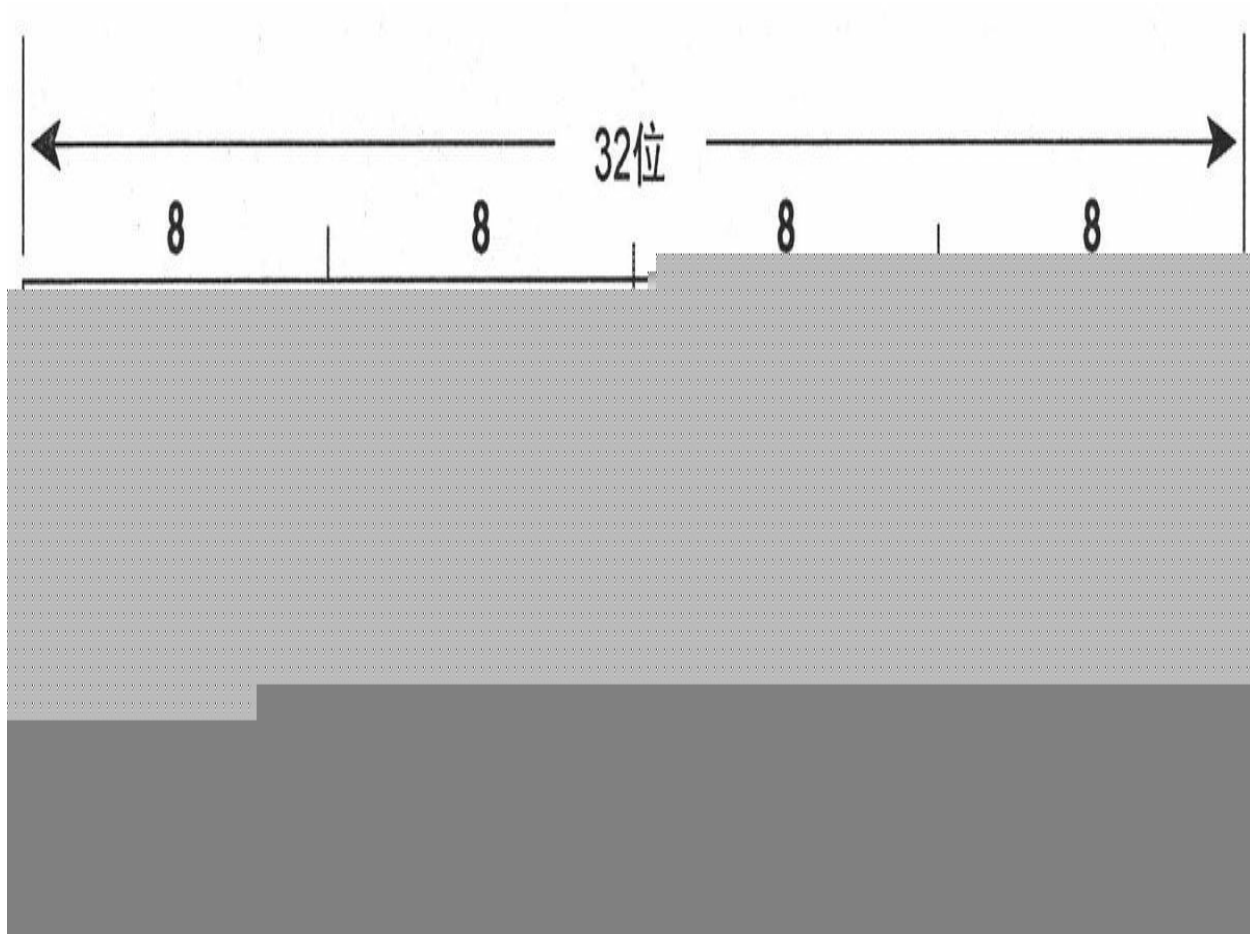


示例3-37 去往目标地址10.1.5.1的数据包在匹配到路由表项10.0.0.0/8之后被转发到下一跳路由器192.168.1.34



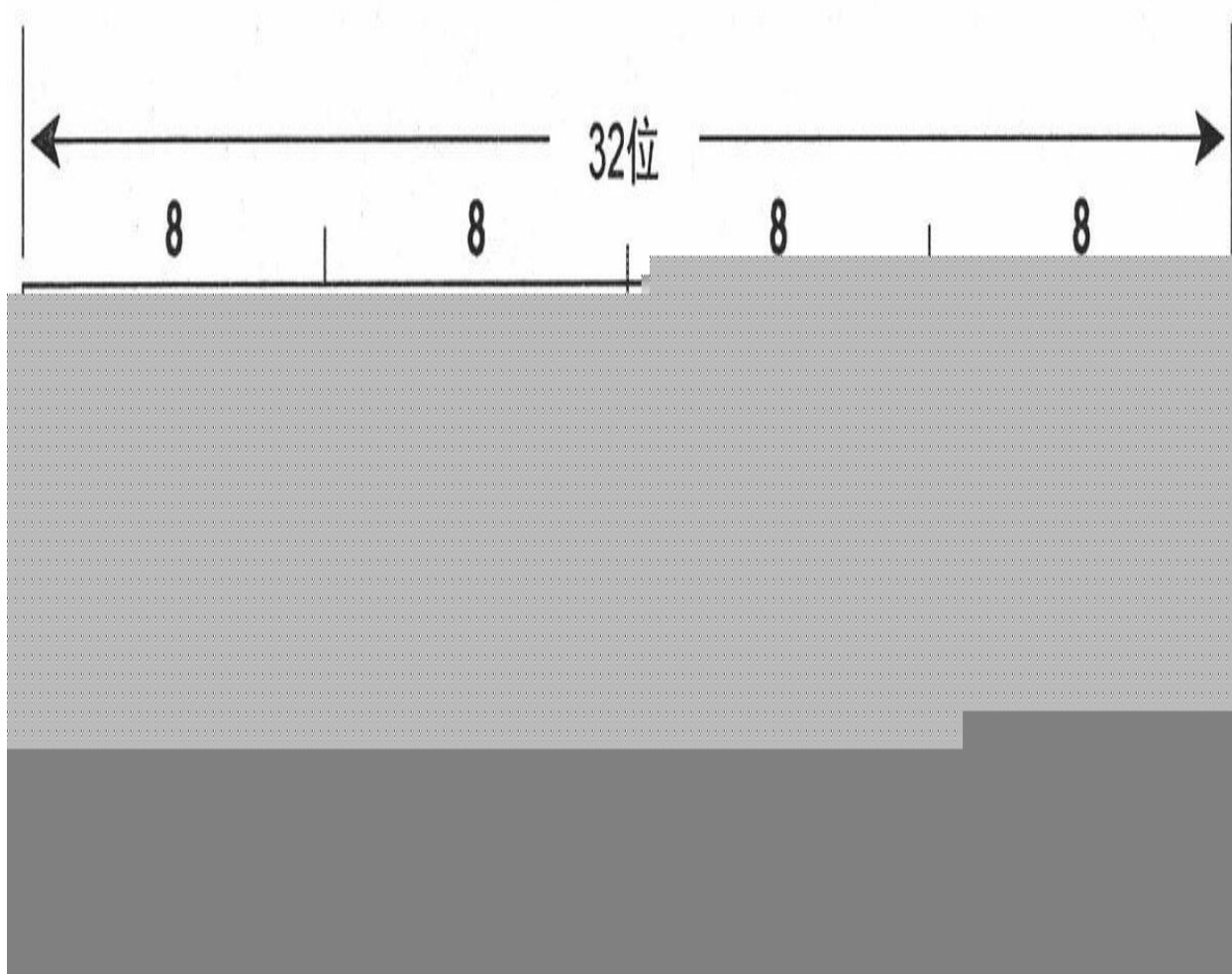
然后，需要检查Eeyore的路由表（参见示例3-38）。目标地址10.1.5.1可以匹配到路由表项10.1.0.0/16，该表项的下一跳地址为10.4.6.1，它是Tigger的一个接口地址。

示例3-38 10.1.5.1在匹配到路由表项10.1.0.0/16之后被转发到10.4.6.1



示例3-39给出了Tigger的路由表。目标地址在匹配到路由表项10.1.0.0/16之后将被转发给Piglet（192.168.1.194）。

示例3-39 10.1.5.1在匹配到路由表项10.1.0.0/16后将被转发到192.168.1.194



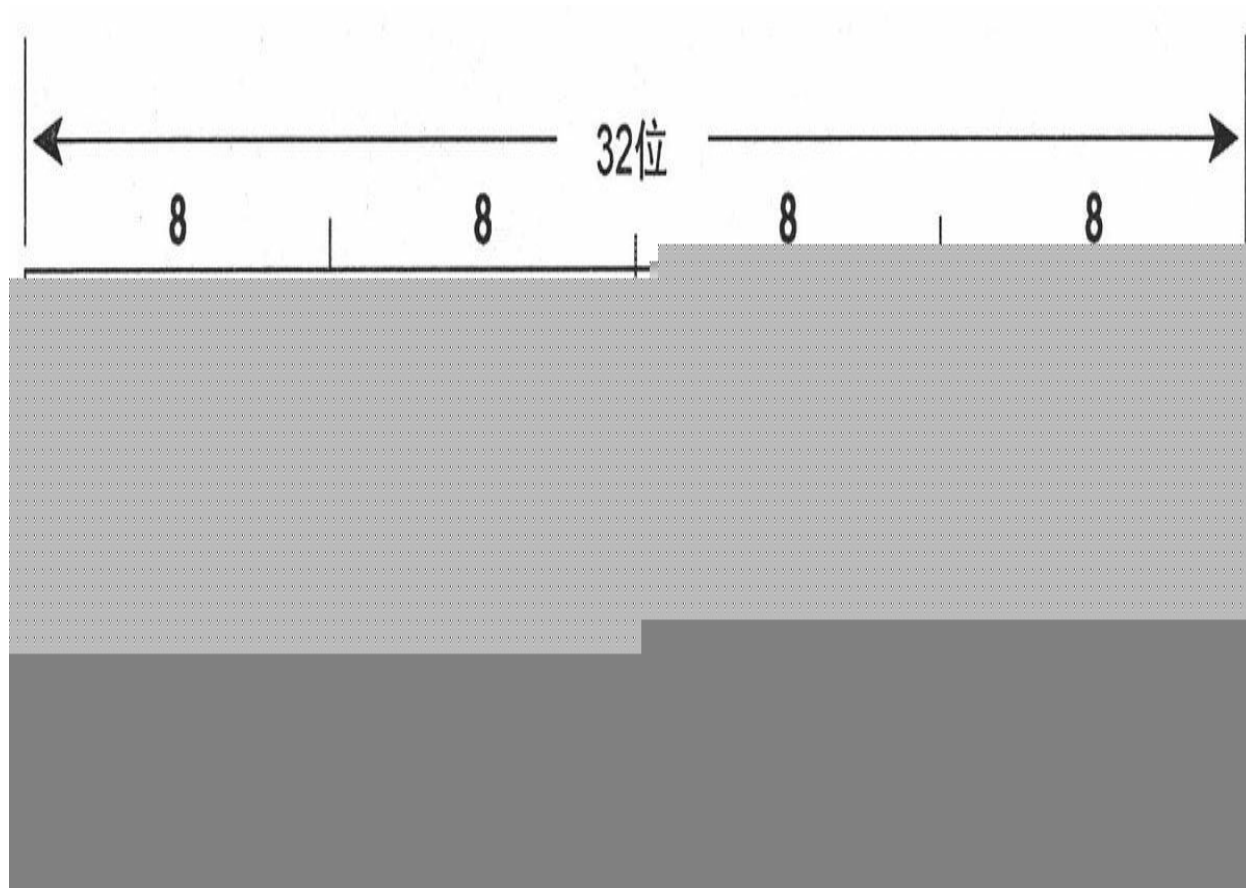
Piglet的路由表（参见示例3-40）显示出目标网络10.1.0.0是一个直连网络。换言之，数据包已经到达目的地了，因为目标地址10.1.5.1就是Piglet自己的接口地址。因为去往目标地址的路径经过验证是正确的，所以我们可以认为来自Pooh的ICMP回应数据包可以到达目标网络。

下一步是跟踪ICMP回应应答数据包所经过的路径。为了跟踪这一路径，读者需要知道回应数据包的源地址——这个地址将是回应应答数据包的目标地址。从路由器发出数据包的源地址就是发送数据包的接口地址。[\[7\]](#)在本例中，最初由Pooh向192.168.1.34转发回应数据包。在图3-9中，显示出数据包的源地址为192.168.1.33。所以Piglet将要发送的回应应答数据包的目标地址就是192.168.1.33。

参考示例3-34中Piglet的路由表，可以发现192.168.1.33将会匹配到路由表项192.168.1.0/24并被转发到192.168.1.193，它是Tigger的另一个接口。重新检查示例3-39中Tigger的路由表，首先使我们想起还有一条关

于192.168.1.0的路由。可是，如果根据这些信息准确地解释那里发生的实际情况，那么需要十分小心。

示例3-40 目标网络10.1.0.0直接连接在Piglet上



比较一下Tigger路由表中有关10.0.0.0子网和192.168.1.0子网的路由表项。10.0.0.0的标题表明其子网大小各不相同；换句话说，指向子网10.4.7.0 Tigger的静态路由使用了24位掩码，指向子网10.1.0.0的静态路由使用了16位掩码。该路由表为每个子网记录了正确的掩码。

192.168.1.0的标题则不同，它表明Tigger知道192.168.1.0有3个子网，且掩码都为255.255.255.224。用这个掩码可以确定目标地址192.168.1.33所属的目标网络为192.168.1.32/27。但是路由表中只有关于192.168.1.64/27、192.168.1.0/27和192.168.1.192/27的路由表项，而没有关于192.168.1.32/27的路由表项，因此路由器不知道该如何到达这个子网。

这样问题就很清楚了，ICMP的回应应答数据包是在Tigger处被丢弃的。一种解决办法是，另外创建一条指向网络192.168.1.32的静态路由，掩码为255.255.255.224，指向下一跳的地址为192.168.1.65或10.4.6.2。还有一种解决办法是，将关于192.168.1.0的路由表项的掩码由255.255.255.224改为255.255.255.0。

此案例的精髓就是当你跟踪路由时，你必须考虑完整的通信过程。

注意：不仅要验证去往目标网络的路由是正确的，还要验证返回的路径也是正确的。

3.3.2 案例研究：协议冲突

如图3-10所示，两台路由器被两个以太网连接起来，其中一个以太网包括一个简单的网桥。该网桥同时还负责处理其他几条没有画出的链路的流量，所以有时网桥会变得十分拥挤。主机Milne是一台承担重要任务的服务器，网络管理员担心网桥会延误Milne的流量，所以在Roo上添加了一条指向Milne的主机路由，该路由指引数据包使用图上方的以太网以避开该网桥。

这种解决方案看上去是合理的，但是实际并非如此。在添加上面那条静态路由后，数据包经Roo不但不能被路由到服务器，而且经Kanga路由的数据包也不能到达服务器，尽管没有对Kanga作过改动。

第一步首先检查路由表。Roo的路由表（参见示例3-41）指明目标地址为172.16.20.75的数据包实际上被转发到Kanga的接口E1上，这正是我们所期望的。由于目标网络直接连接到Kanga上，所以不再需要进一步的路由。经过快速检查确定在Kanga和Milne上的两个以太网接口都工作正常。

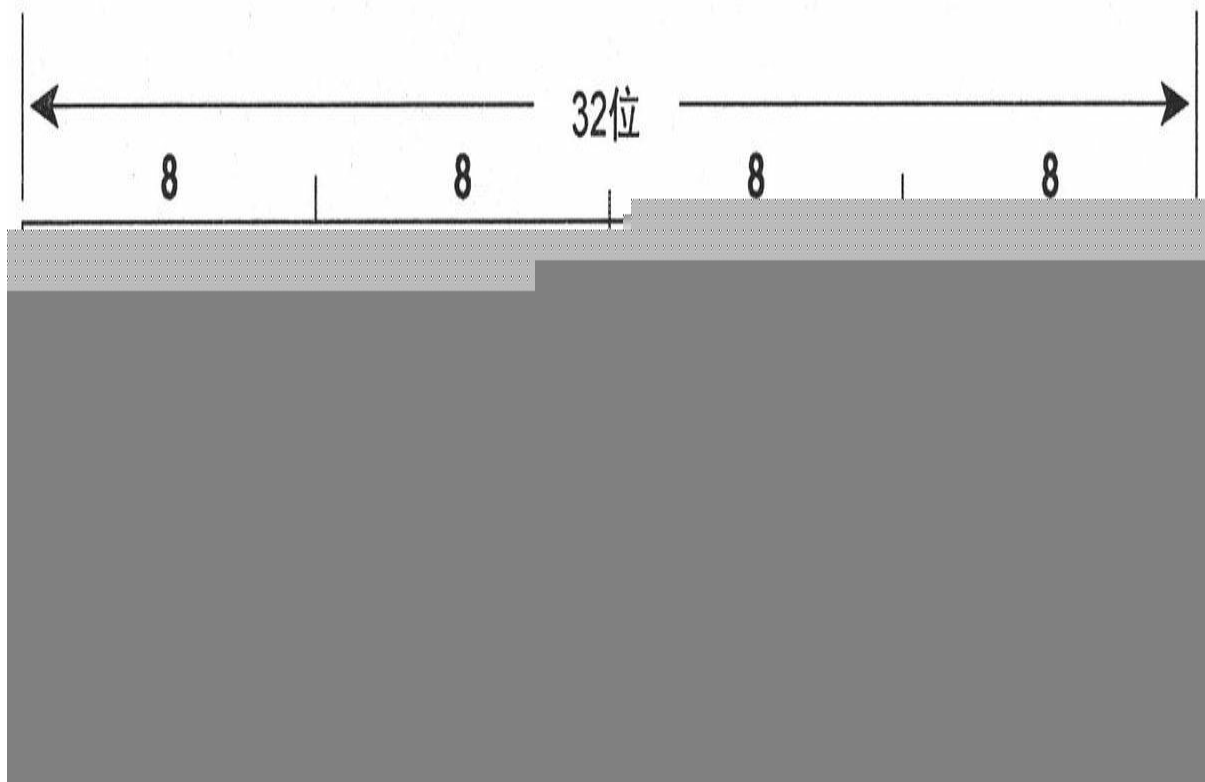
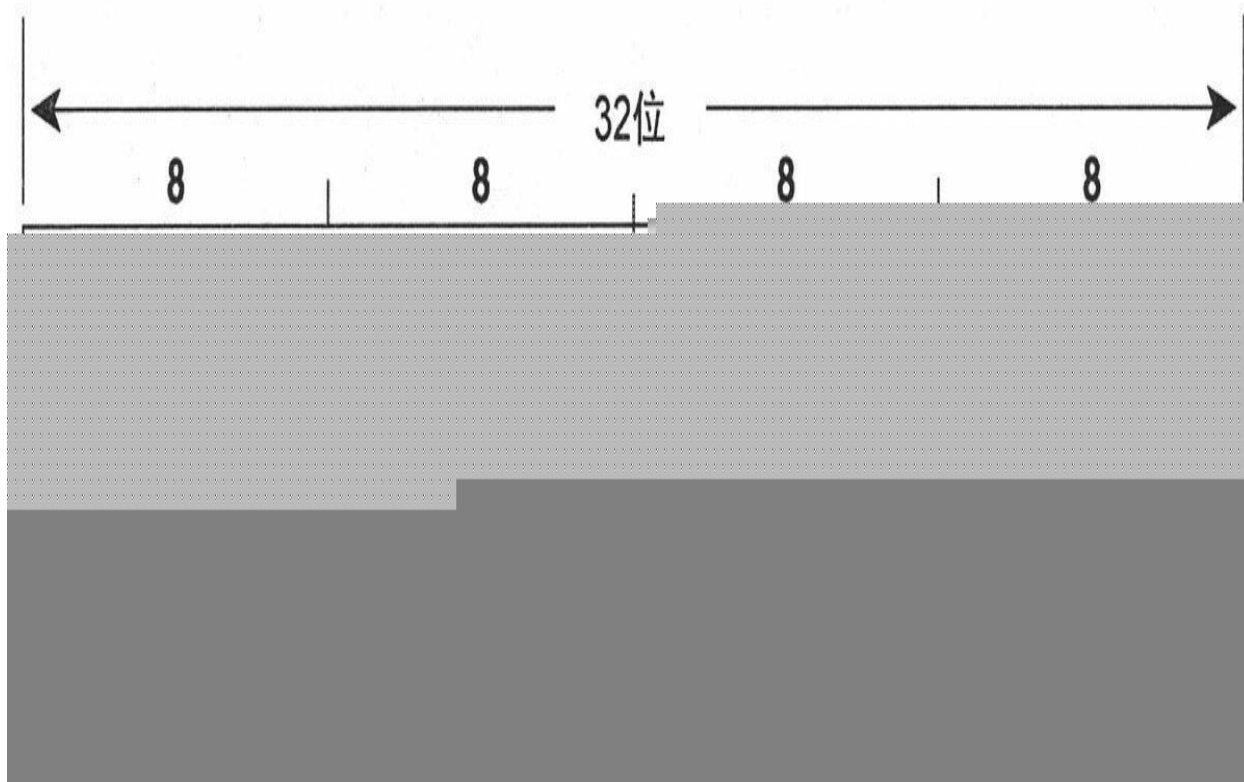


图3-10 主机路由指引从Roo到Milne的数据包经过上面的以太网，这样可以避开偶尔发生拥塞的网桥

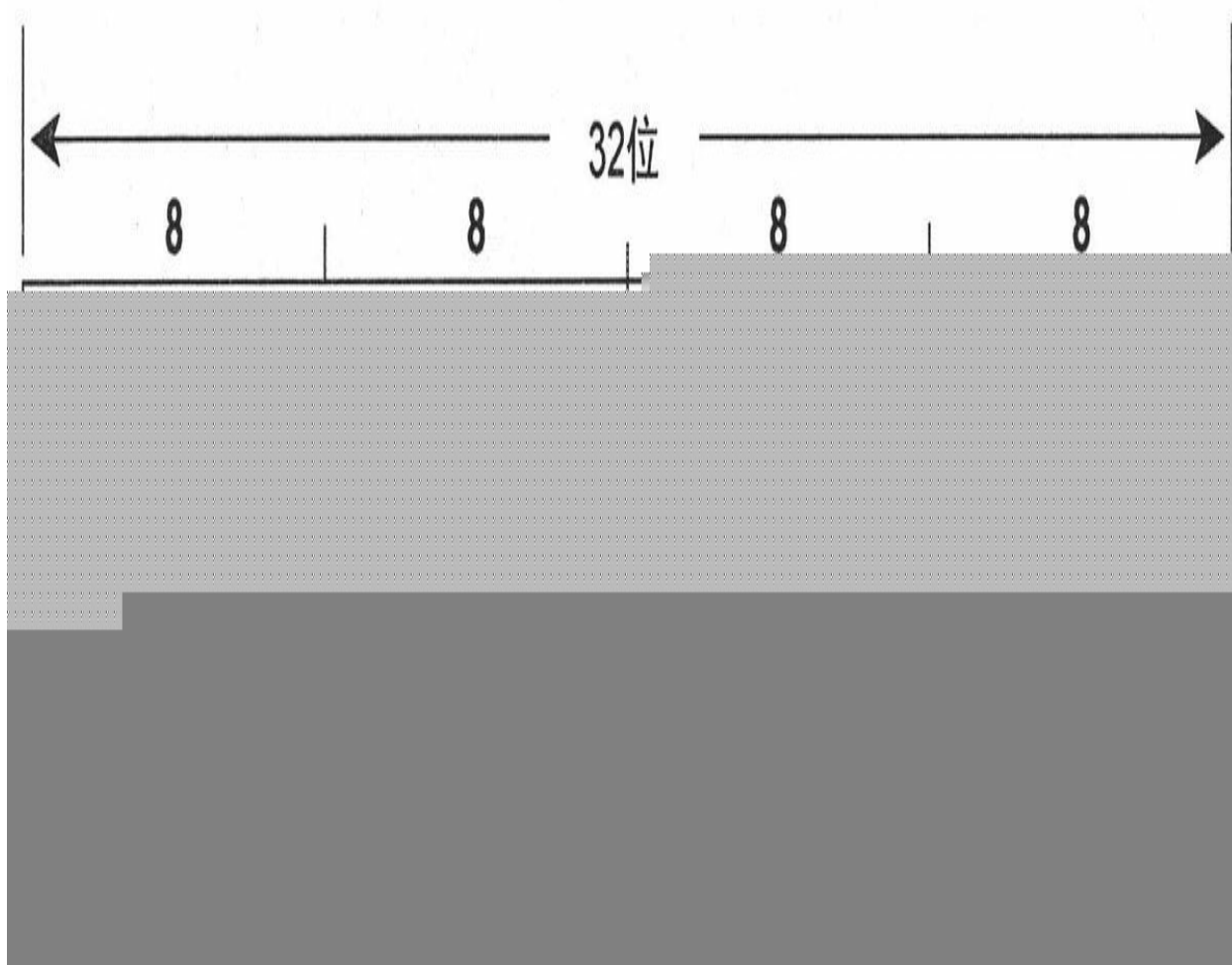
示例3-41 在Roo的路由表中显示有一条经Kanga接口E1指向Milne的静态主机路由



在示例3-42中，从Roo向Milne执行跟踪命令，发现以下症状。Kanga应该发向Milne的数据包被发向Roo的接口E0。Roo又将数据包发给Kanga接口E1，接着Kanga再次将数据包发回给Roo。这看上去像是发生了路由选择环路，但是为什么呢？

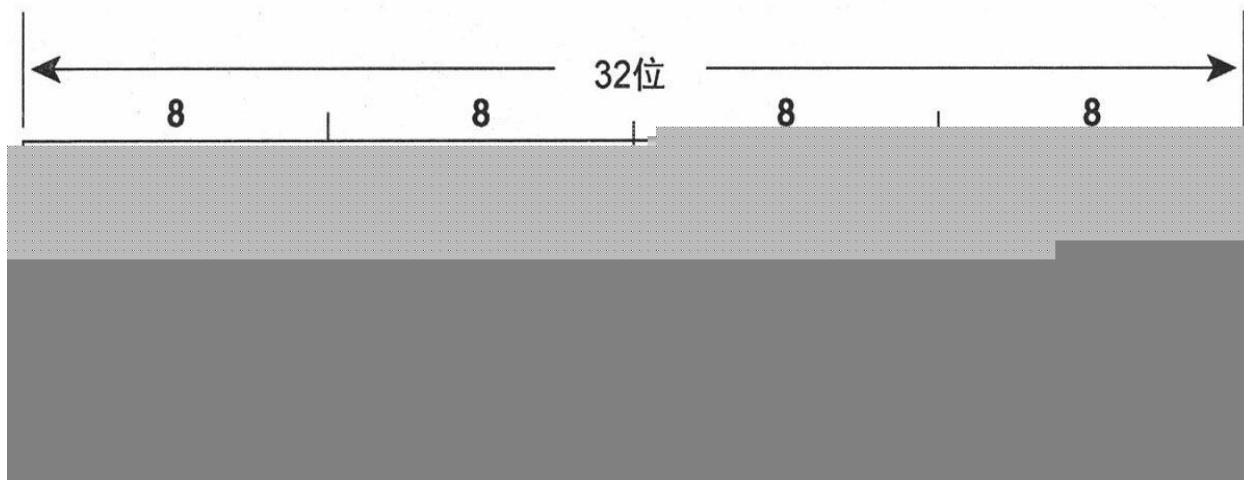
此故障的可疑之处在于Kanga不应该对数据包进行路由选择，但事实恰恰相反。Kanga应该能够识别出数据包的目标地址属于它的直连网络172.16.20.0，然后使用数据链路向主机传送数据包。因此疑点发生在数据链路上。路由器是否具有经过某条逻辑路径到达目标网络的正确信息，这一点我们可以通过查看路由表来获知。同样的，我们应该检查ARP高速缓冲区，来确定路由器是否具有经过某条物理路径到达某台主机的正确信息。

示例3-42 从Roo到Milne进行跟踪，发现Kanga将本应发往正确目的地的数据包回发给了Roo



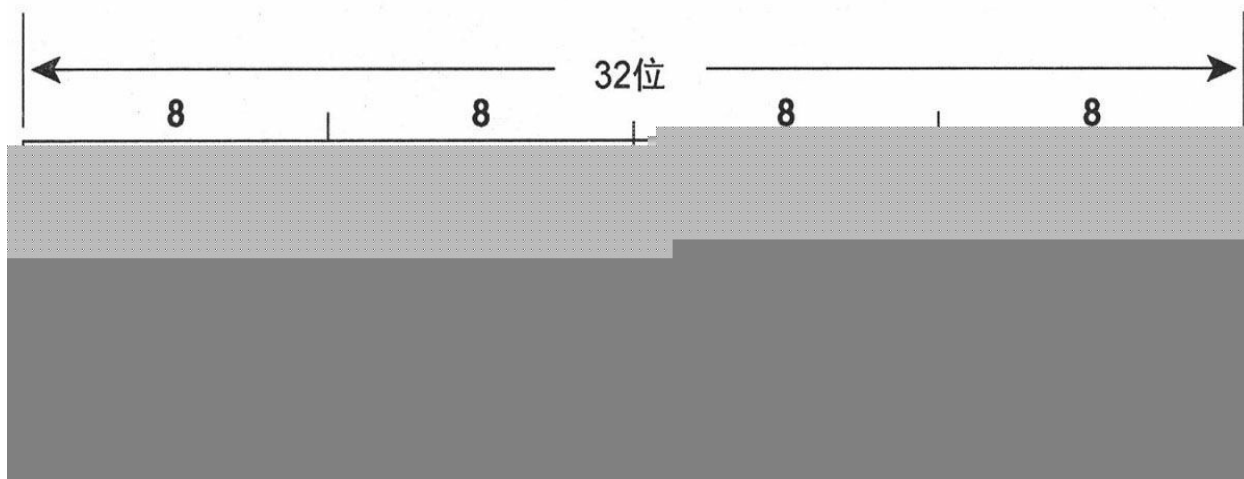
示例3-43给出了Kanga的ARP高速缓冲。在Kanga的高速缓冲中，Milne的IP地址与MAC标识符00e0.1e58.dc39相对应。但是当检查Milne的接口时，发现Milne的MAC标识符为0002.6779.0f4c，因此断定Kanga一定获取了不正确的信息。

示例3-43 在**Kanga**的**ARP**高速缓冲中，有一个关于**Milne**的表项，其中数据链路标识是错误的



再次查看Kanga的ARP高速缓冲，令人感到疑惑的是与Milne相对应的MAC标识符类似于Kanga自己Cisco接口的MAC标识符（路由器接口的MAC地址没有相应的存活时间）。因为Milne不是Cisco产品，所以MAC标识符的前3个字节应该与Kanga接口的MAC标识符不同。网络上其他的Cisco产品惟有Roo，因此应该检查一下Roo的ARP高速缓冲（参见示例3-44）。经查Roo接口E0的MAC标识符即为00e0.1e58.dc39。

示例3-44 Roo的ARP高速缓冲显示出Kanga获取的Milne的MAC标识符实际上是Roo的接口E0的



所以Kanga错误地认为Roo的接口E0就是Milne的接口。Kanga使用00e0.1e58.dc39作为发向Milne数据帧的目标标识符，Roo接收到该帧，在读取封装数据包的目标地址之后，又将数据包路由回Kanga。

但Kanga是如何得到这个错误信息的呢？答案是代理ARP。当Kanga首次收到发往Milne的数据包时，它将发送ARP请求，该请求将询问Milne的数据链路标识符。Milne发回了响应，但是Roo也在接口E0收到了此ARP请求。由于在Roo上也有一条通向Milne的路由，这条路由所在的网络不是Roo收到ARP请求的网络，所以Roo发送了一个代理ARP应答。Kanga收到Milne的ARP应答后将相关信息输入到ARP高速缓冲内。由于网桥的时延造成了来自Roo代理ARP的应答随后到达Kanga，这时Kanga用新信息覆盖了ARP缓冲内的原始信息。

解决该问题的办法有两种。一种是使用以下命令关闭Roo E0接口上的代理ARP功能：

```
Roo(config)#interface e0
```

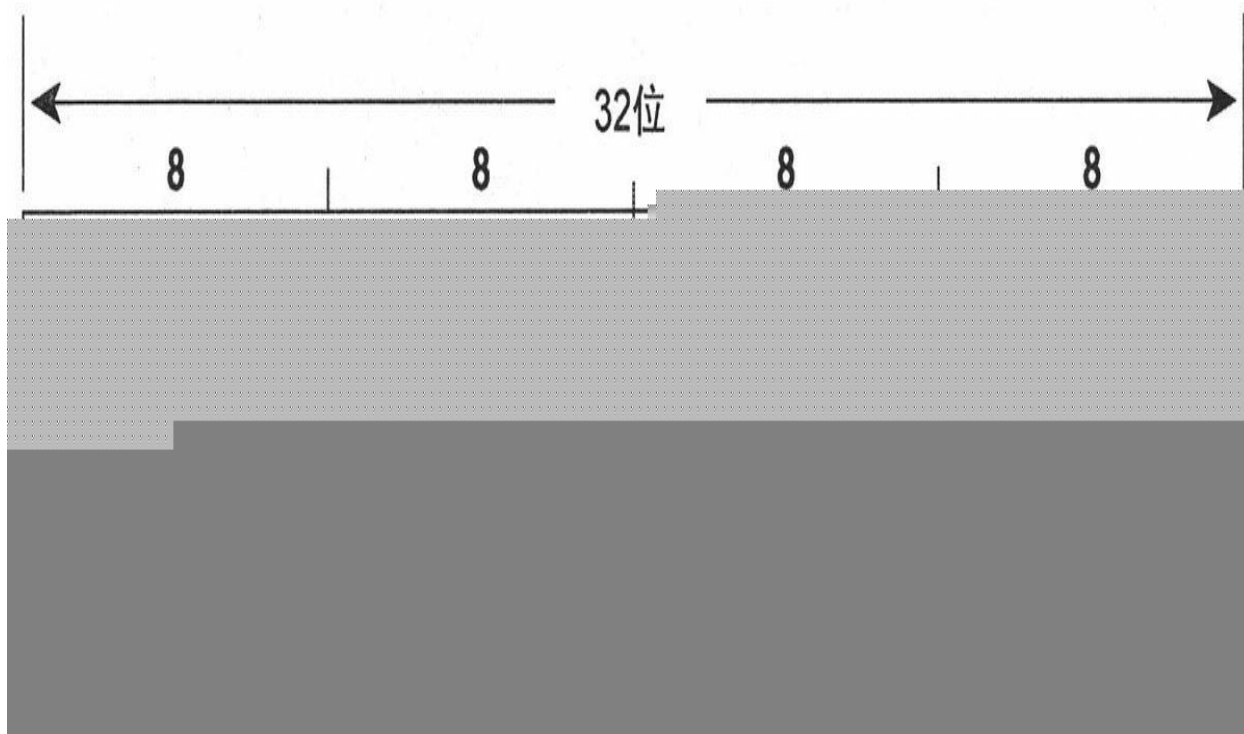
```
Roo(config-if)#no ip proxy-arp
```

第二种办法是在Kanga上为Milne配置静态ARP表项：

```
Kanga(config)#arp 172.16.20.75 0002.6779.0f4c arpa
```

该表项不会被任何ARP应答所覆盖。示例3-45显示出正在输入静态ARP表项以及Kanga上ARP高速缓冲的相应结果。

示例3-45 静态ARP表项纠正了代理ARP所造成的错误



两种方法哪一种最好取决于网络环境。使用静态ARP表项意味着如果Milne的网络接口被更换，那么需要修改相应的ARP表项来反应新的MAC标识符。另一方面，如果没有主机使用代理ARP功能，关闭此功能也是一种很好的办法。

[3.3.3 案例研究：被取代的路由器](#)

图3-11中有两台路由器Honeypot和Honeybee通过帧中继链路相连，路由器被配置为IPv4和IPv6，并且使用静态路由建立了静态路由表。

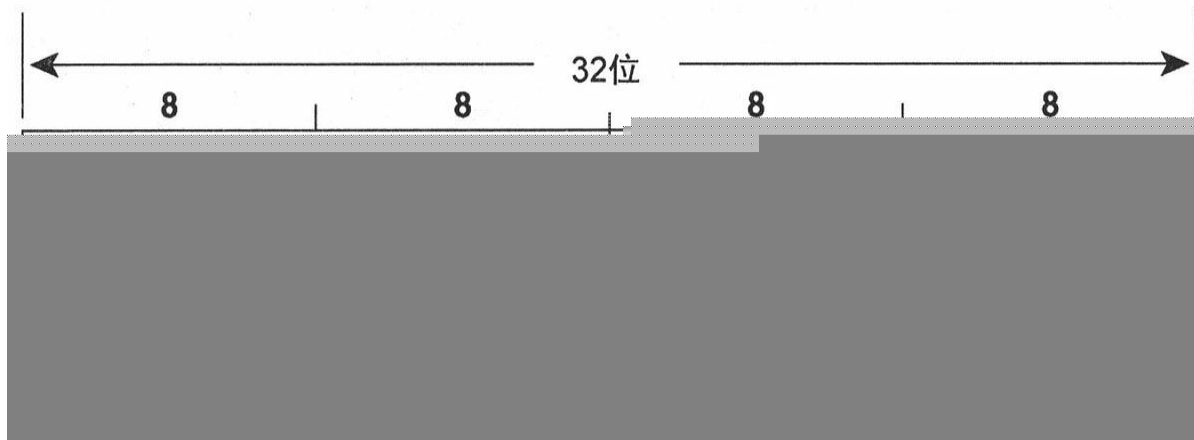
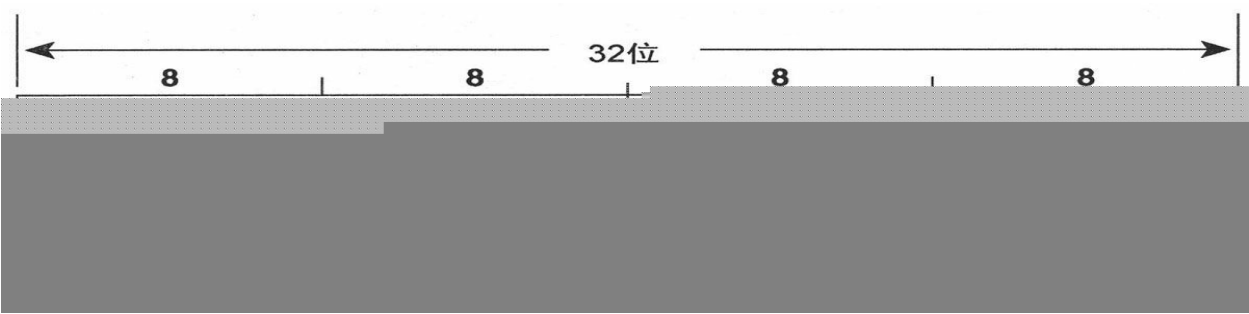


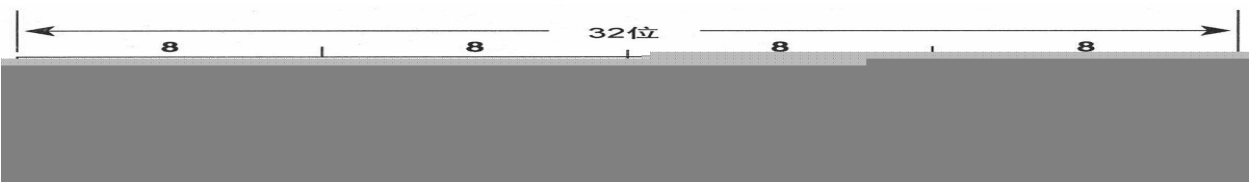
图3-11 同时使用IPv4和IPv6的简单网络

示例3-46和示例3-47分别给出了Honeypot和Honeybee的路由配置。

示例3-46 Honeypot的路由配置

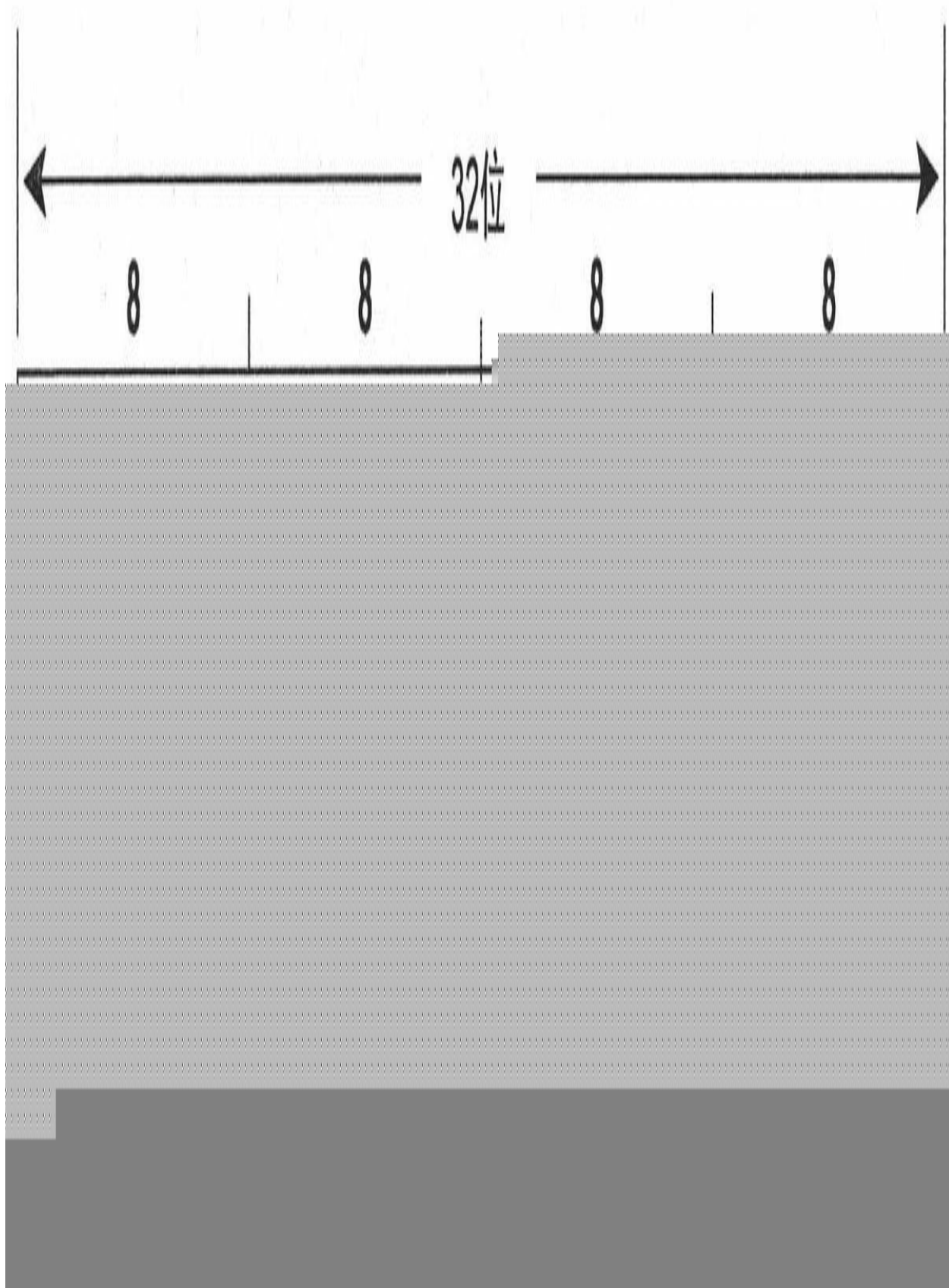


示例347 Honeybee的路由配置



如示例3-48所示，在两台路由器之间进行ping和trace测试，结果显示网络工作正常。而且两个站点之间的流量也正常。

示例3-48 Pings和traces的结果显示两台路由器之间可以成功地进行通信

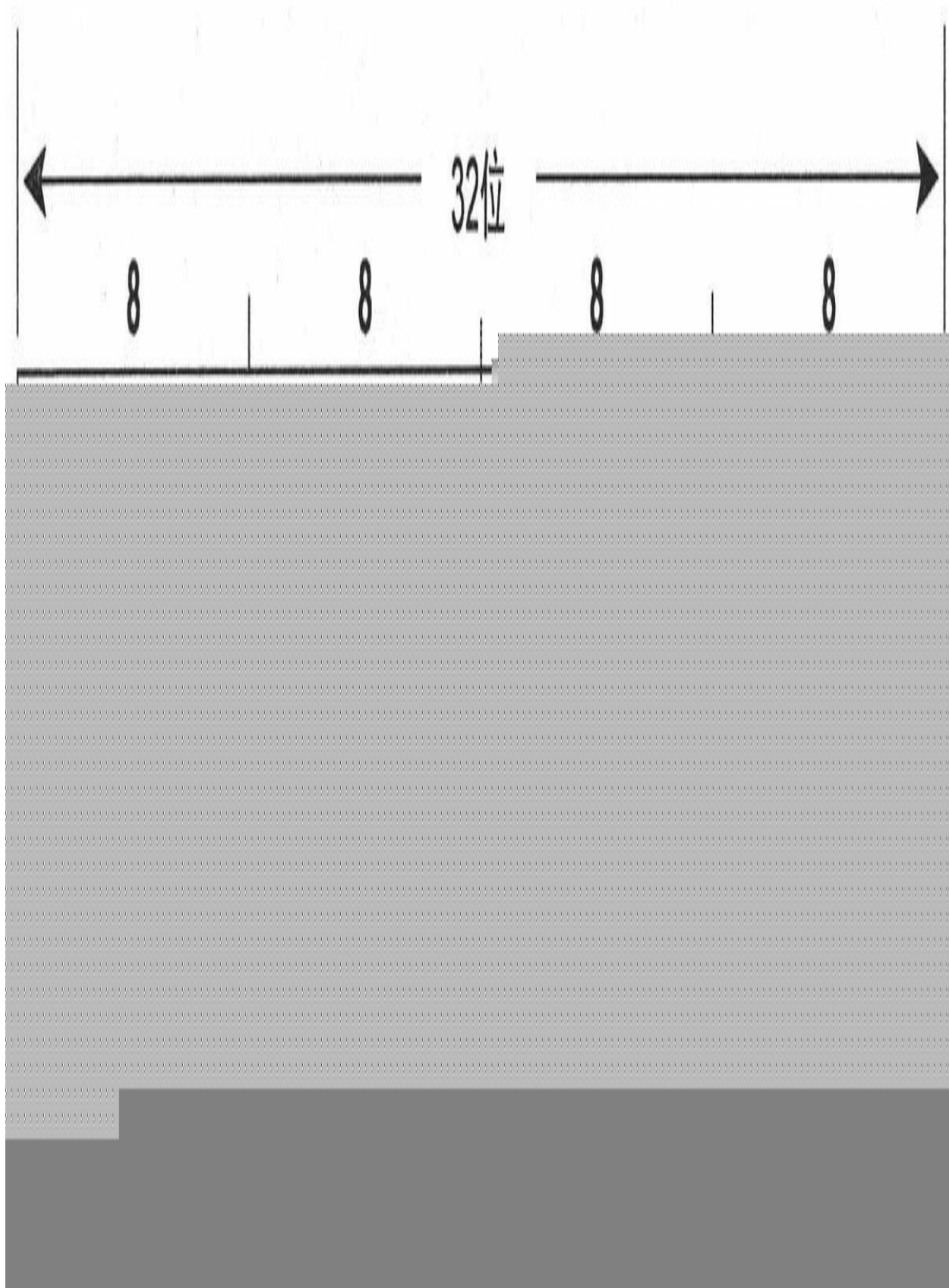


Honeypot可以从自己的以太网地址到达Honeybee的以太网地址，IPv4和IPv6都可以。Honeypot还可以通过Honeybee到达FEC0:0:0:A::1。

可是Honeybee被一台新的路由器替换了，当把Honeybee的配置导入新路由器时，所有的接口都工作正常。两站点之间的测试结果显示IPv4工作正常，但是IPv6发生故障（参见示例3-49）。

示例3-49 在路由器被替换后，**IPv4**工作正常，但是**IPv6**却无法工作

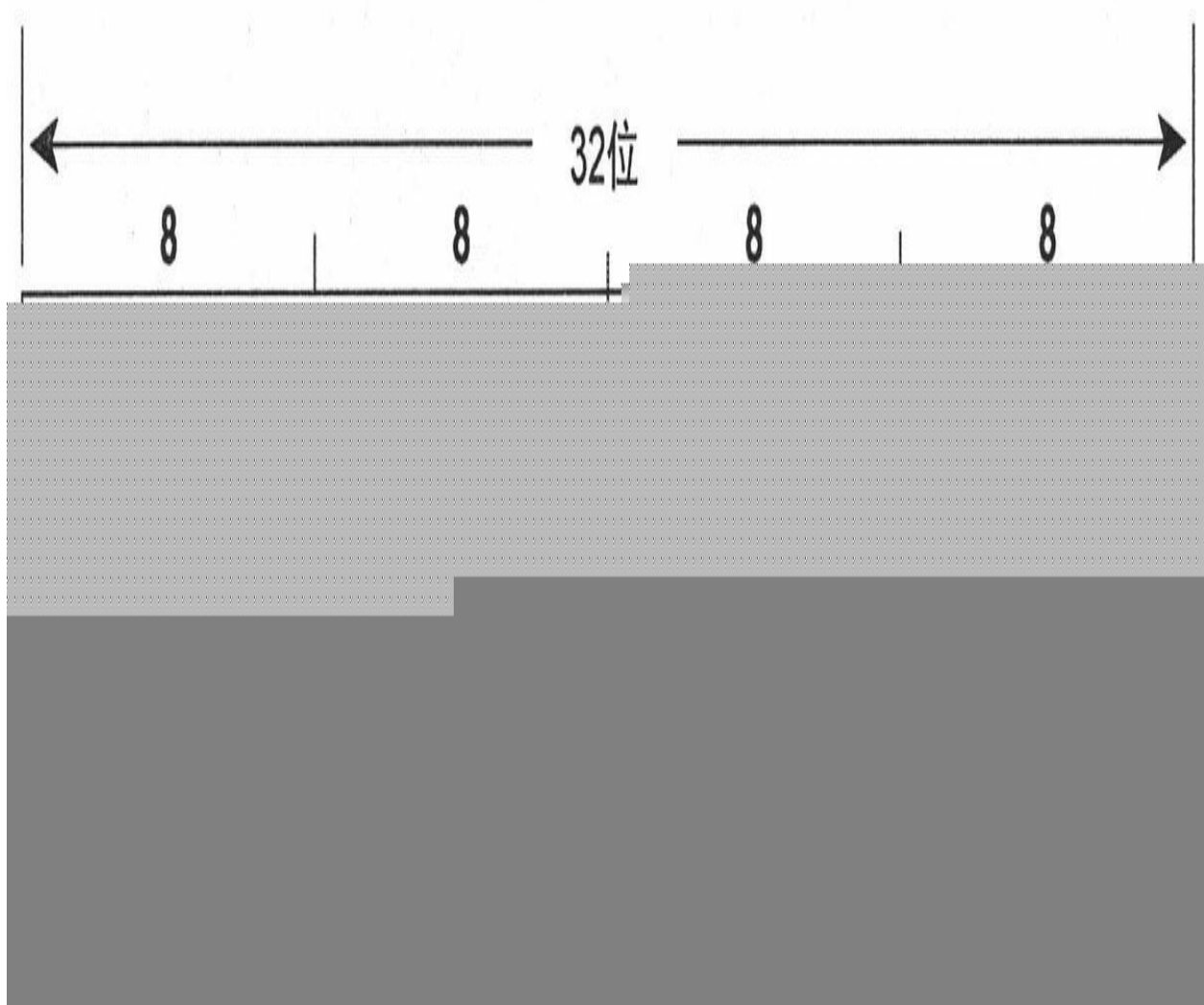




向Honeybee以太网的IPv6地址FEC0::8:204:c1FF:FE50:F1C0 ping和trace都告失败，但是向FEC0::A:0:0:0:1trace仍然是成功的。

让我们看一下示例3-50中Honeypot的路由表。IPv6的路由表中有一条路由通过下一跳地址FEC0::3:204:C1FF:FE50:F1C0指向FEC0:0:0:8::/62。

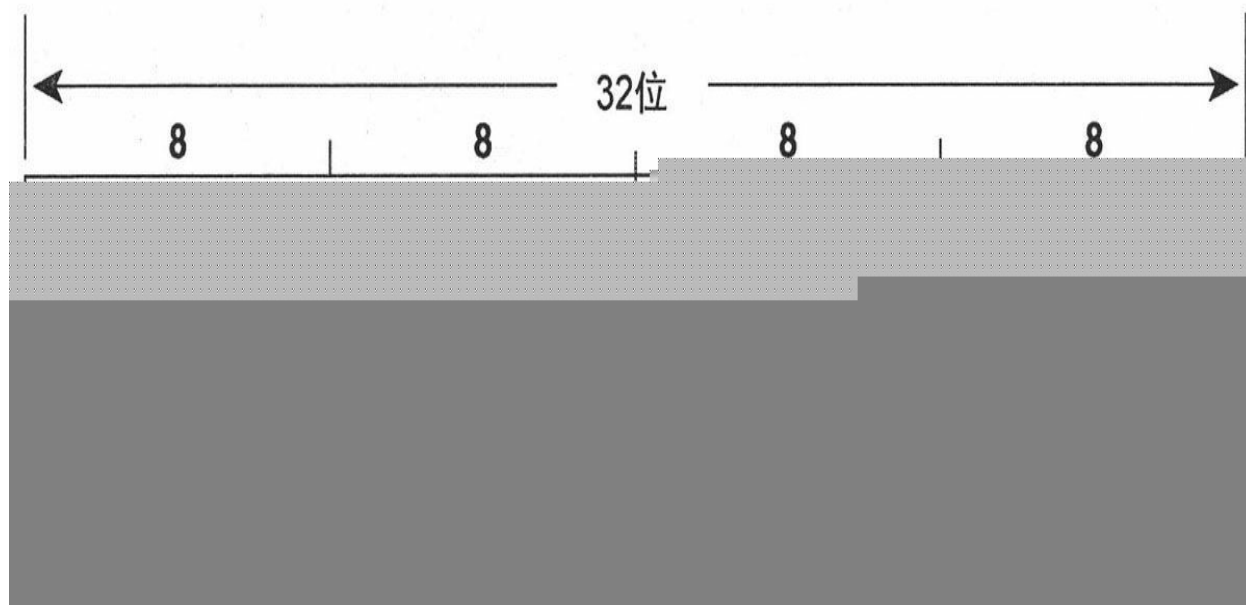
示例3-50 IPv6的路由表没有发生变化，并且安装了静态路由



从示例3-50可以知道，通过FEC0::3:204:C1FF:FE50:F1C0可以到达FEC0:0:0:8::/62，并且FEC0:0:0:3::/64被连接到接口Serial0/0.2上；去往FEC0:0:0:8::/64和FEC0:0:0:A::/64的流量都会从该接口转发给Honeybee。

替代操作前后的路由是完全一样的。那么问题出在哪里呢？或许Honeybee的以太网接口没有正常工作，这可以查看一下示例3-51中Honeybee的以太网接口状态。

示例3-51 命令**show ipv6 interface Ethernet 0/0**可以显示接口的状态以及有关**IPv6**的一些配置信息

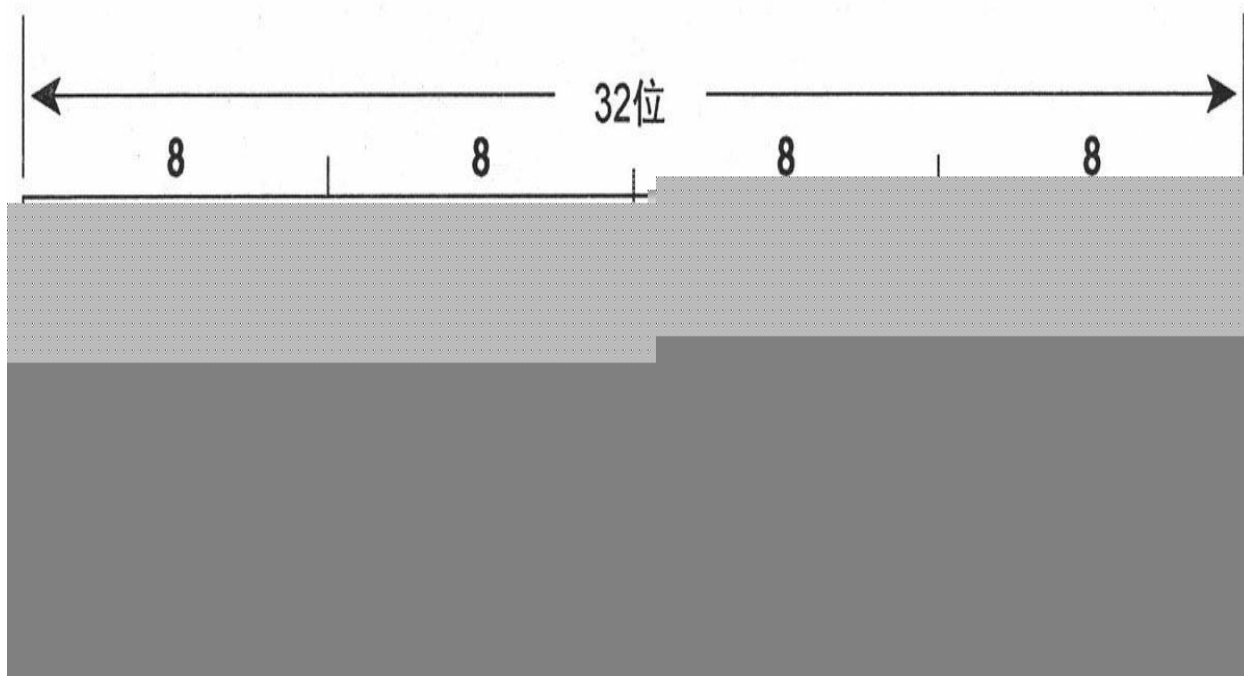


从示例3-51中可以看到接口状态正常并且配置了IPv6。根据网络的文档和图表，该接口在替换前可以被ping通，但是替换后就不行了。

如果我们仔细查看以太网接口的输出信息，可以发现全球单播地址的后64位发生了变化。现在地址变为FEC0::8:2B0:64FF:FE30:1DE0。因为IPv6地址是EUI-64格式，所以后64位由接口的MAC地址确定。当路由器被替换后，MAC地址也随之改变。路由器上不再有FEC0::8:204:C1FF:FE50:F1C0存在。这就是ping不通的原因。

但是根据路由表（参见示例3-50），对于目标网络FEC0:0:0:8::/62，Honeypot的静态路由仍然指向FEC0::3:204:C1FF:FE50:F1C0。如果MAC地址发生了变化，那么这个地址就不是同步接口的地址了。示例3-52给出了Honeybee同步接口的新地址。

示例3-52 在路由器上，所有配置**EUI-64**地址的同步接口地址后**64**位都是相同的，它来自路由器上的一个**MAC**地址



如果目标地址在FEC0:0:0:8::/62的范围内，Honeypot的IPv6静态路由将把相关流量指向下一跳地址FEC0::3:204:C1FF:FE50:F1C0。虽然下一跳地址是原来路由器的MAC地址，然而使用该路由的流量仍然可以被成功路由。

这里虽然指定的下一跳地址不再是合法的，但是它和新的合法地址在相同的子网中（FEC0:0:0:3::/64），正像我们在Honeypot的路由表中看到的那样，这个子网被连接到Honeypot的接口Serial0/0.2上。通过对转发表的递归查询，虽然指定的下一跳地址不存在，但是去往FEC0:0:0:A::1的流量依然从Serial0/0.2送出。

每当路由器或接口卡被替换时，一定要记得修改参考旧路由器EUI-64格式IPv6地址的路由表项。

3.3.4 案例研究：追踪IPv6故障路由

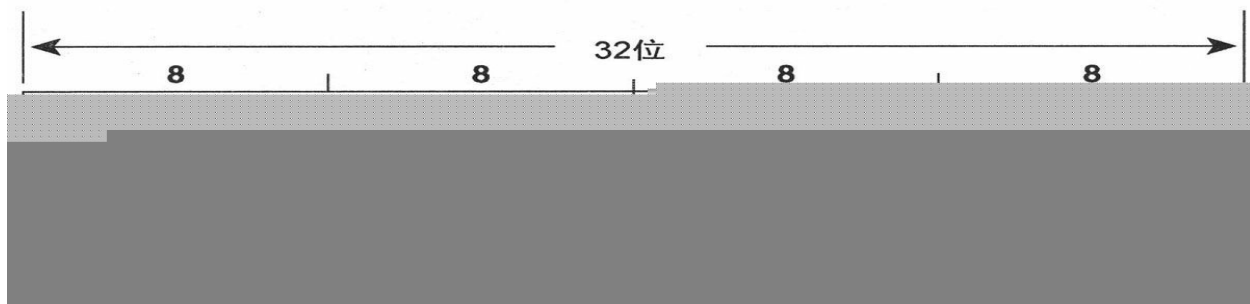
在图3-3中，在Honeypot和Honeytree之间添加了一条链路，当主链路万一发生中断时，它可以作为备份链路。改动后的新网络如图3-12所示。由于使用该网络的IPv6应用对时延不敏感，但是对带宽却要求很高，所以网络管理员决定使用新添加的链路作负载均衡，因此在每台路由器上添加静态路由。



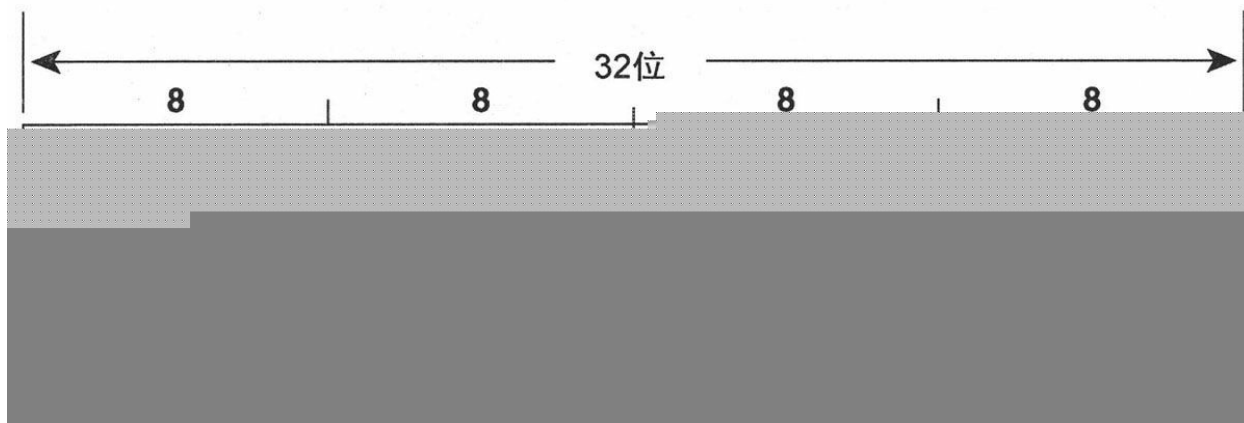
图3-12 在IPv6网络中添加了另一条可供选择的路径，网络变成三角形网络，使得路由器之间可以相互到达

如示例3-53所示，对应于Honeypot每个已存在的路由，示例中又添加了第二条路由。示例3-54和示例3-55也分别对Honeytree和Honeybee进行了类似的操作。

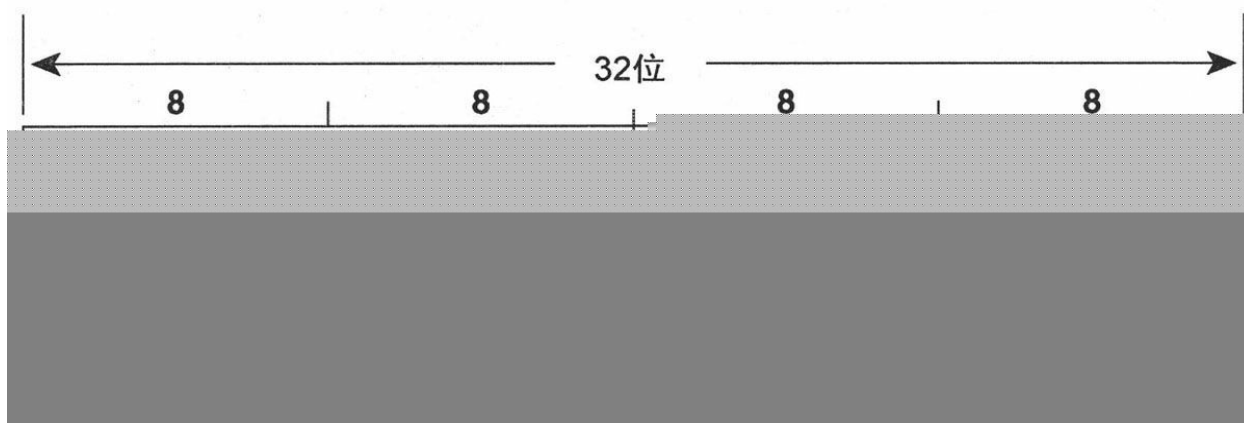
示例3-53 对于图3-12的新网络，**Honeypot**的相应配置



示例3-54 对于图3-12的新网络，**Honeytree**的相应配置

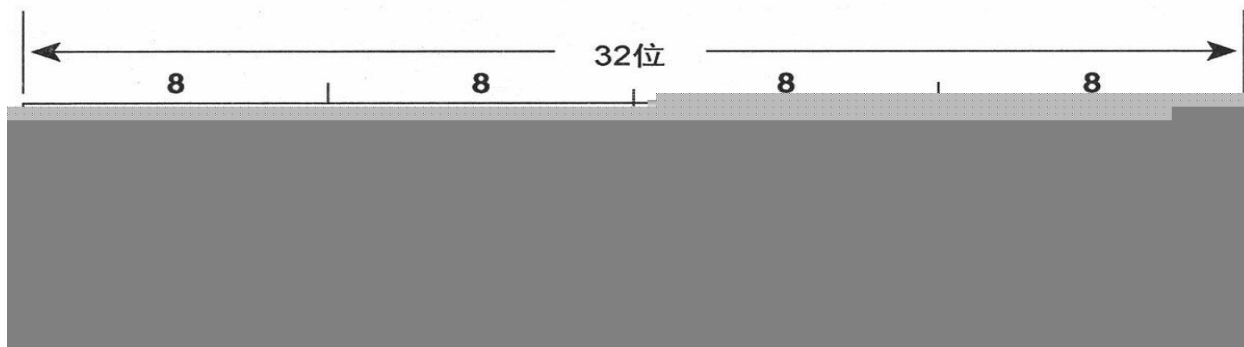


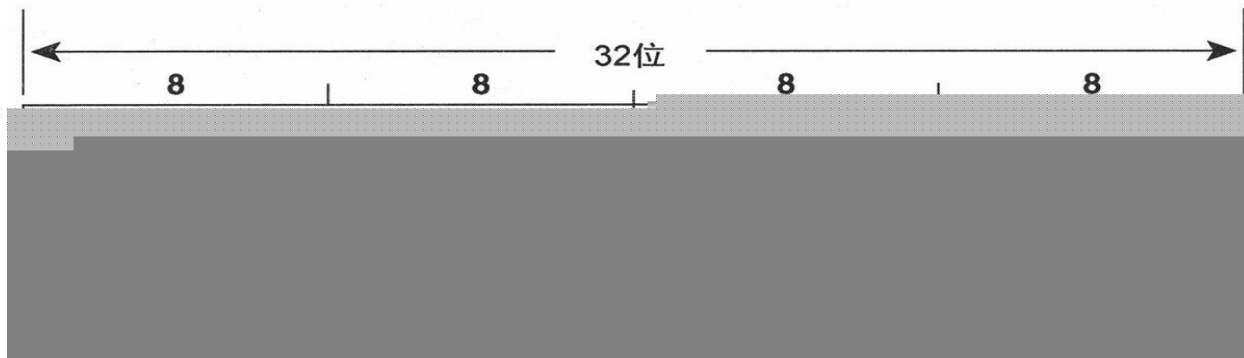
示例3-55 对于图3-12的新网络，Honeybee的配置



现在，IPv6的路由选择间歇性的出现故障，有时工作，有时不工作。可以看出路由表中的静态路由与设计完全相同。从Honeypot向Honeybee的以太网接口进行ping测试，有时成功，有时出现主机不可达（参见示例3-56）。

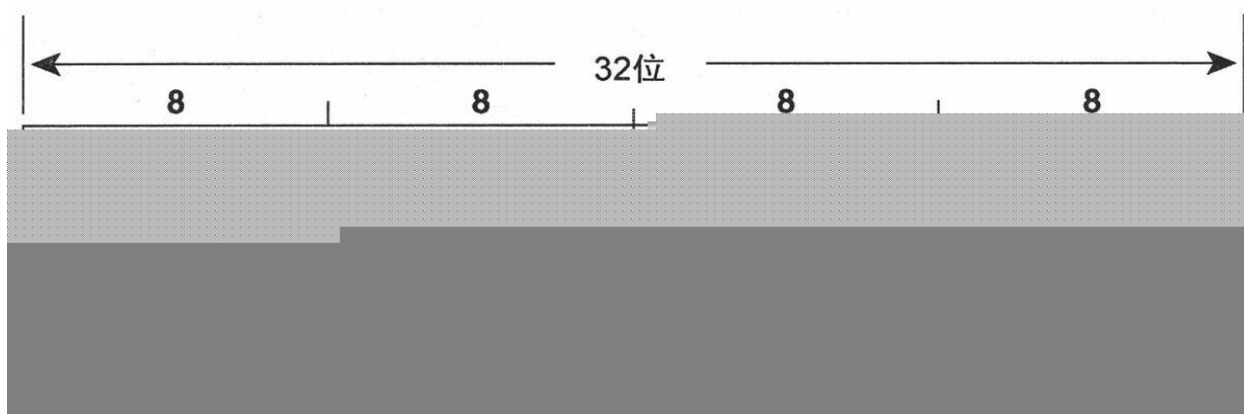
示例3-56 IPv6有时可以ping通





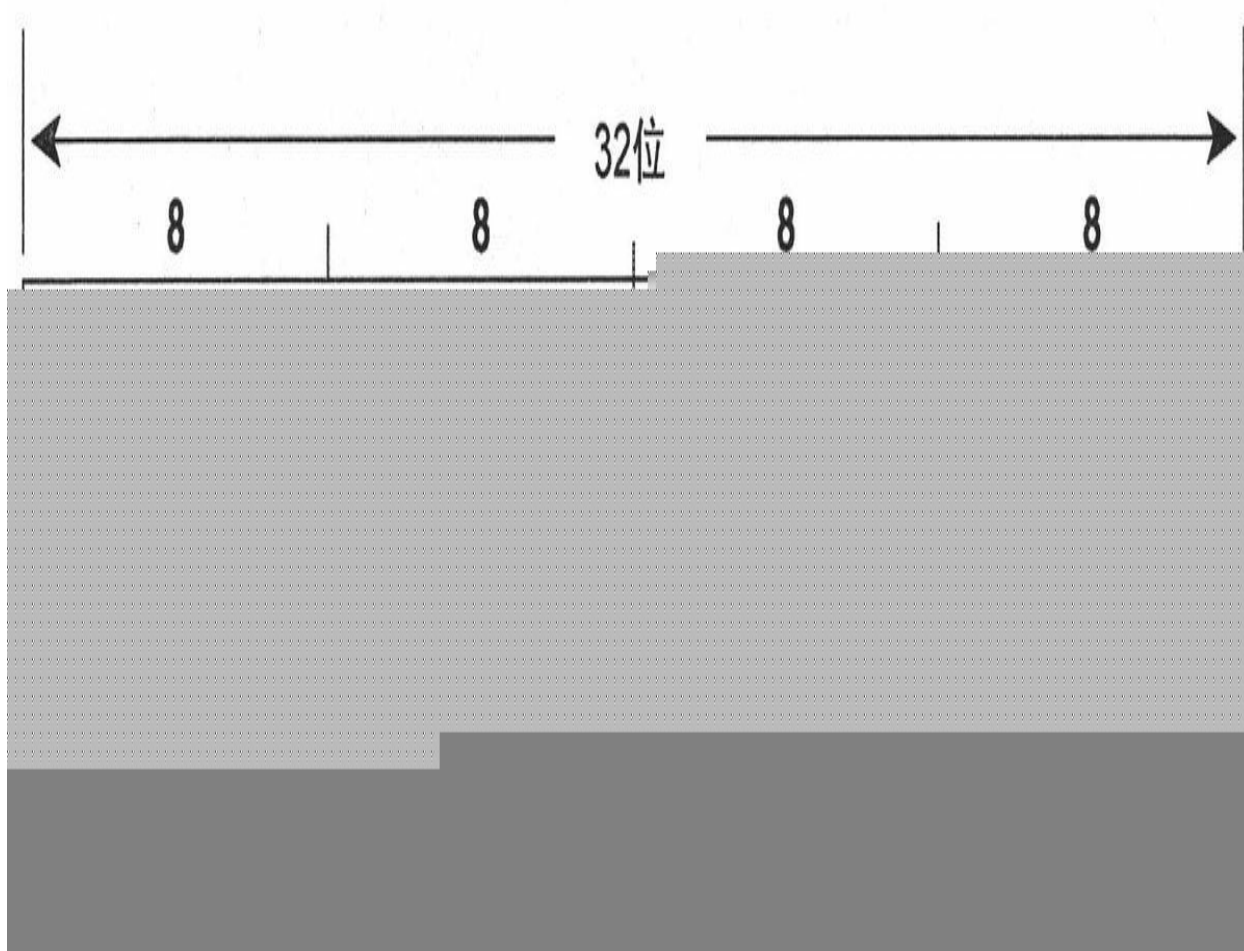
在Honeypot上调试IPv6的ICMP数据包，你可以发现能成功地收到一些响应数据包（参见示例3-57），但是还会收到一些来自Honeytree的目标不可达的ICMP消息。

示例3-57 在Honeypot上的调试结果显示：有时成功，有时失败



在Honeytree上调试ICMPv6的结果（参见示例3-58）显示，IPv6的数据包正在从S0/0.3和S0/0.2到达。

示例3-58 调试结果显示数据包从两个不同的接口到达



源地址为fec0::2:230:94ff:fe24:b780的数据包从接口S0/0.3到达Honeytree（来自Honeypot的数据包经过Honeypot至Honeytree的链路到达）。对于目标网络fec0::8:204:c1ff:fe50:flc0来说，出站接口是S0/0.2或S0/0.3。这是因为Honeytree执行基于数据包的负载均衡，在S0/0.2和S0/0.3之间交替转发数据包。当数据包从S0/0.2出站并且从S0/0.3入站时，数据包将会被转发。但是如果从S0/0.3出站，而又从S0/0.3入站，则路由器会产生ICMP错误信息，表明ping失败。

如果数据包从同步接口出站后又从相同的接口入站，那么就表明存在路由环路。由于路由器正在处理交换和基于数据包的负载均衡，每台路由器都会轮流使用两条路由，所以一些数据包会从出站的接口又入站。

3.4 展 望

对于精确地控制网络的路由选择行为来说，静态路由不失为一个强有力的工具。然而，如果经常发生网络拓扑变化，那么手动配置方式导致静态路由的管理工作根本无法进行下去。动态路由选择协议能够使网络迅速并自动地响应网络拓扑的变化。在研究特定IP路由选择协议的细节之前，我们首先需要研究一些围绕动态协议的常见问题。第4章将介绍动态路由选择协议。

3.5 总结表：第3章命令总结

arp <i>ip-address hardware-address</i>	把IP类型（别名）地址静态地映射到硬件地址
debug ip packet	显示有关接收、生成、转发IP数据包的信息。而关于快速交换的数据包信息将不被显示
debug ipv6 packet	显示有关接收、生成、转发IPv6数据包的信息
ipcef	为IPv4 启用Cisco 急速转发功能
ip load-sharing {per-packet per-destination}	在出站接口上配置负载均衡方式
ip proxy-arp	启用代理ARP 功能
ip route <i>prefix mask</i> { <i>address</i> \interface[<i>next-hop-address</i>]} [<i>distance</i>][<i>permanent</i>] [<i>name naem</i>] [<i>tag tag-number</i>]	向路由表添加静态路由
ip route-cache	在接口上为IPv4 配置交换高速缓冲的类型
ipv6 cef	为IPv6 启用Cisco 急速转发功能。IPv4 的CEF功能必须在IPv6 CEF 功能之前打开
Ipv6 unicast-routing	启动IPv6 路由选择。缺省情况下IPv6 路由选择是关闭的
Ipv6 route <i>prefix/prefix length</i> { <i>address</i> interface[<i>next-hop-address</i>] } [<i>distance</i>] [<i>permanent</i>] [<i>name name</i>] [<i>tag tag-number</i>]	静态添力口IPv6路由
show cdp neighbor detail	显示邻居路由器的IOS 版本号以及IPv4 和IPv6 接口配置等信息
showip cef	显示CEF转发高速缓冲消息，包括CEF未启用消息

show ipv6 cef {*interface*} detail

显示接口上CEF功能是否被打开

show ipv6 interface {*interface*}

显示接口以及IPv6特定的接口信息

show ip route

显示**IP** 路由表

show ipv6 route

显示IPv6路由表

show ipvb cef

3.6 复习题

1. 路由表中需要保存哪些信息？
2. 当路由表指明对一个地址进行了变长子网划分时，这意味着什么？
3. 什么是非连续子网？
4. 使用什么IOS命令检查IPv4路由表？
5. 使用什么IOS命令检查IPv6路由表？
6. 在路由表中与非直连路由相关的括号内的两个数字表示什么？
7. 当使用出站接口代替静态路由中的下一跳地址时，路由表会有什么不同？
8. 什么是汇总路由？在静态路由选择的上下文中，汇总路由怎样起作用？
9. 什么是管理距离？
10. 什么是浮动静态路由？
11. 等价均分负载与非等价均分负载之间有什么不同？
12. 接口上的交换模式怎样影响均分负载？
13. 什么是递归表查询？

3.7 配置练习

1. 如图3-13所示的网络，为每台路由器配置静态路由。要求每个子网都有独立的表项。
2. 使用最少的路由表项重新配置练习1中的静态路由^[8]（提示：RTA仅有两条静态路由）。
3. 如图3-14所示的网络，为每台路由器配置静态路由。假设所有链路的介质相同。为保证最大利用率和线路冗余，请使用负载均衡和浮动静态路由技术。

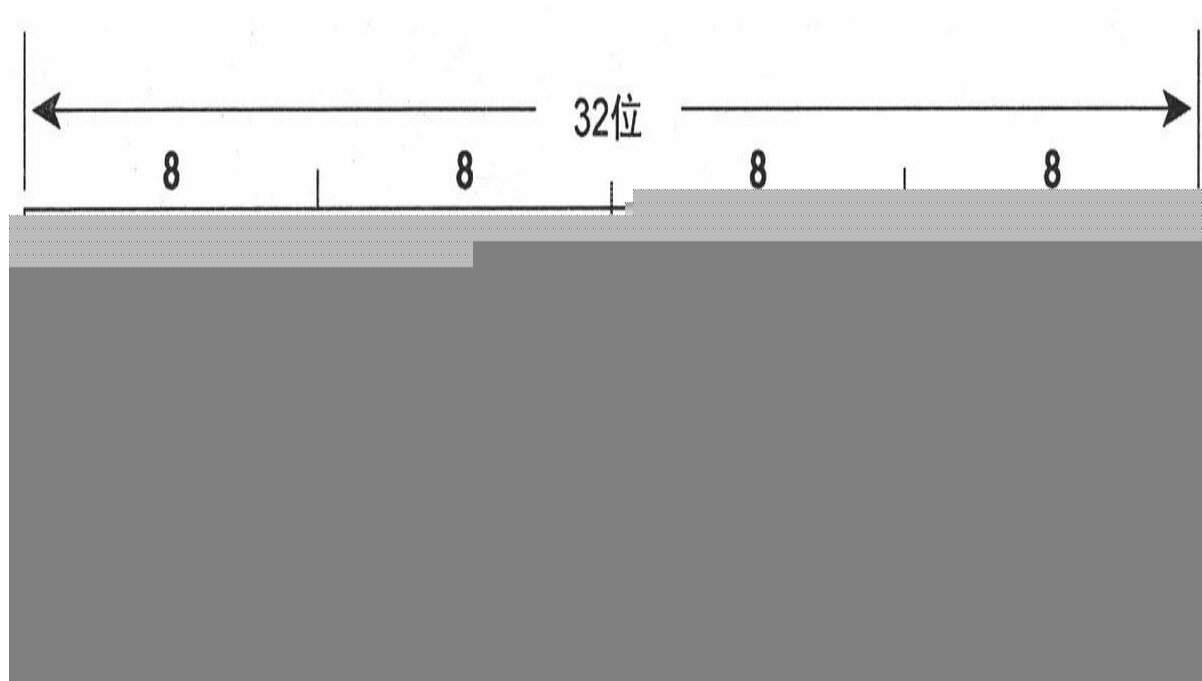


图3-13 配置练习1和练习2中用到的网络

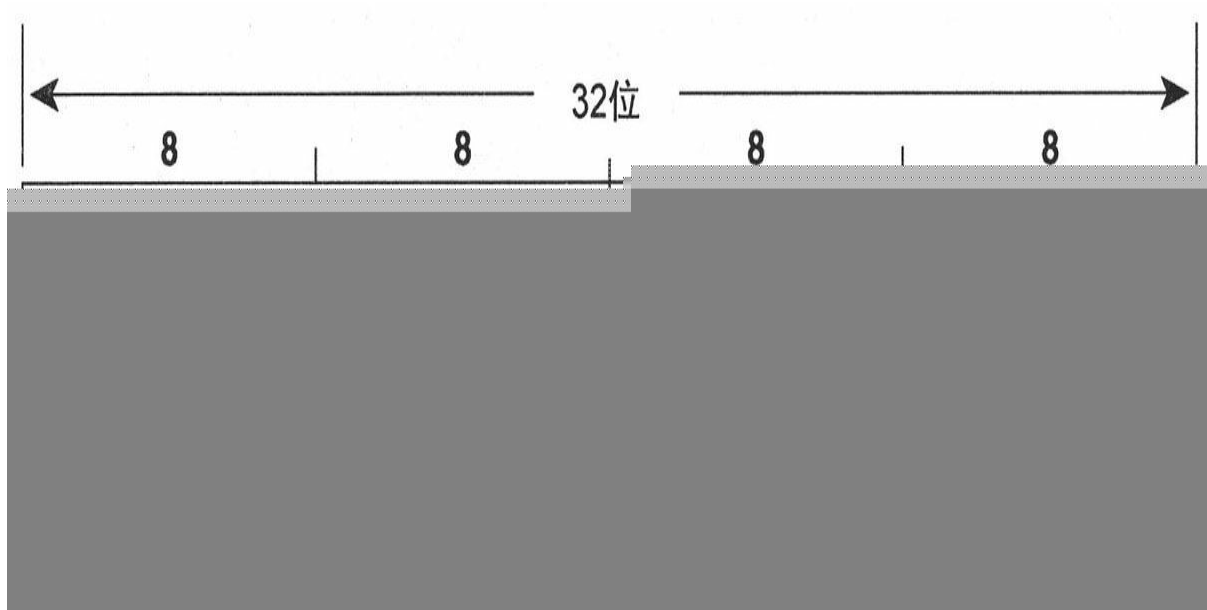


图3-14 配置练习3用到的网络

3.8 故障诊断练习

1. 在图3-2所示的网络和相关配置中，把Piglet的路由配置由

```
Piglet(config)# ip route 192.168.1.0 255.255.0 192.168.1.193
```

```
Piglet(config)# ip route 10.4.0.0 255.255.0.0 192.168.1.193
```

修改为：

```
Piglet(config)# ip route 192.168.1.0 255.255.255.224 192.168.1.193
```

```
Piglet(config)# ip route 10.0.0.0 255.255.0.0 192.168.1.193
```

会发生什么情况？

2. 示例3-59给出了图3-15中路由器的静态路由配置。

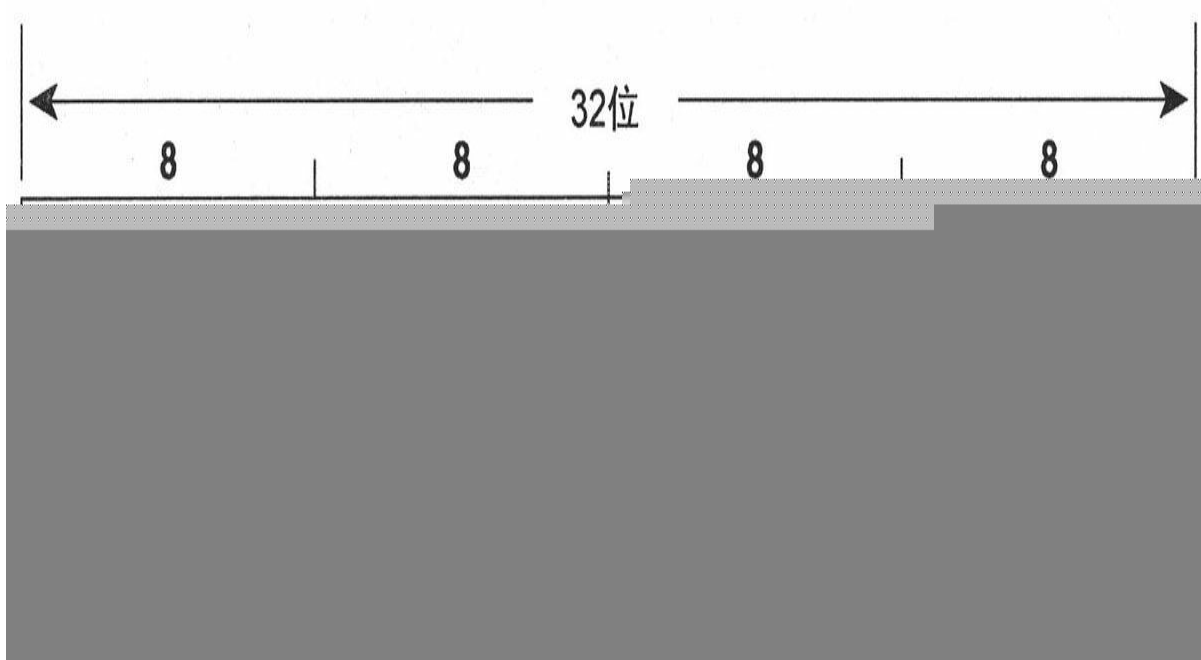
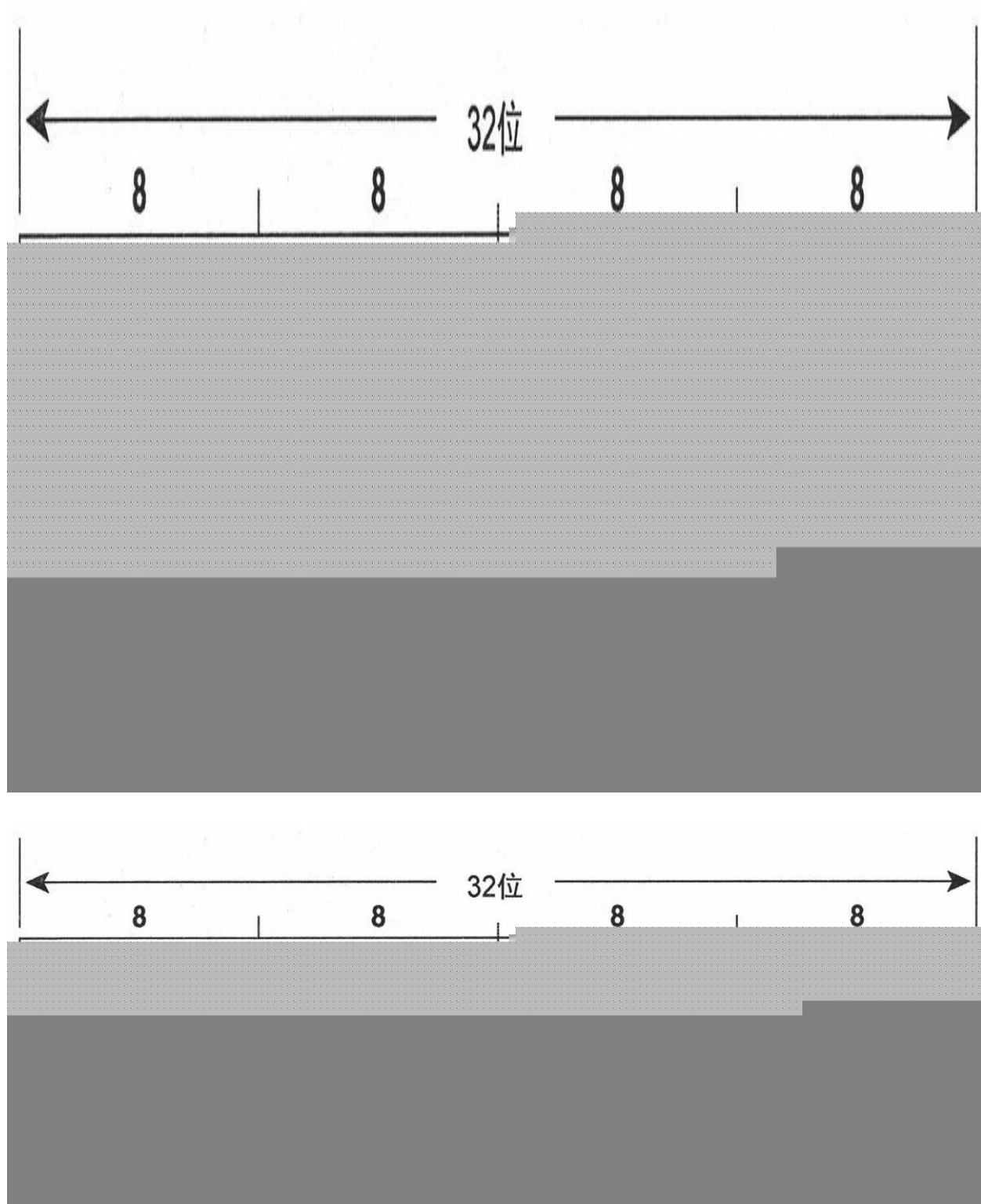


图3-15 故障诊断练习2用到的网络

示例3-59 图3-15中路由器的静态路由配置



用户抱怨网络中存在一些连通性问题。请在静态路由配置中找出错误。

3. 图3-16给出了另一个网络，同样有用户抱怨存在连通性问题。示例3-60到示例3-63分别给出了4台路由器的路由表，请找出静态路由配置的错误。

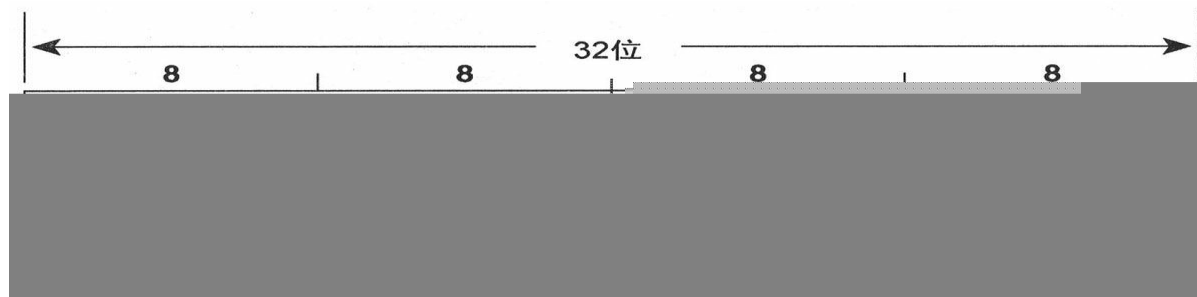
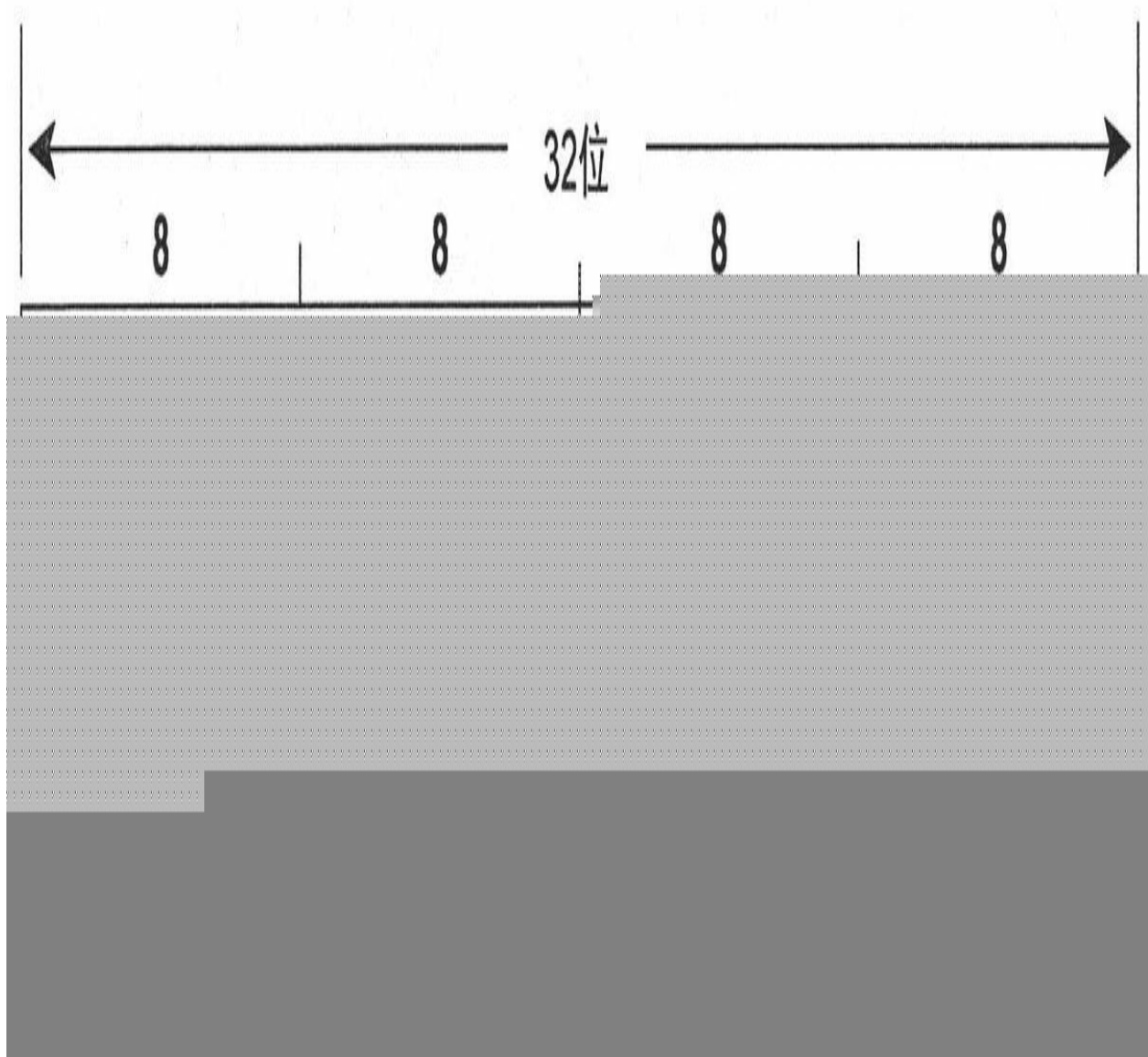
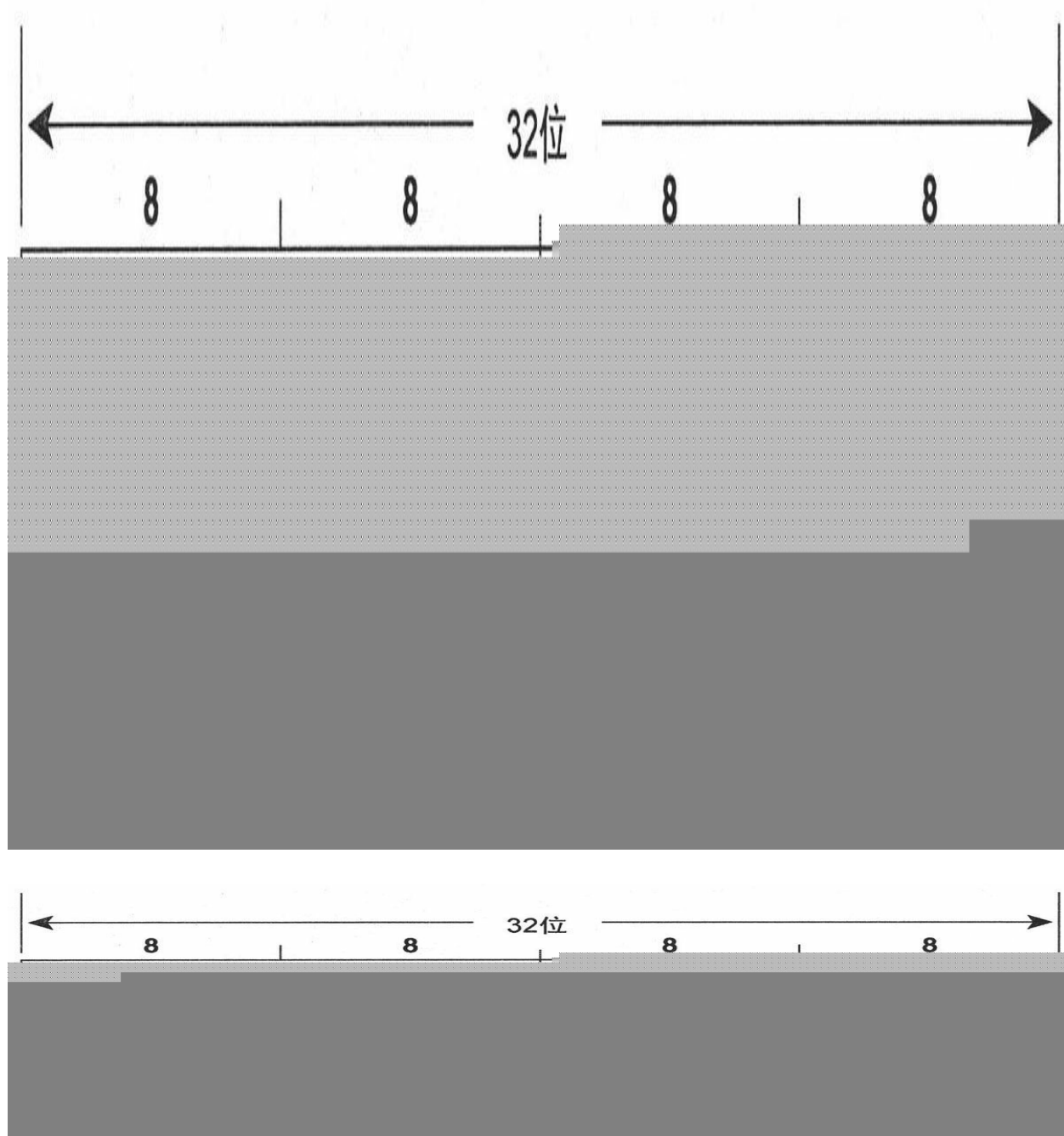


图3-16 故障诊断练习3的网络

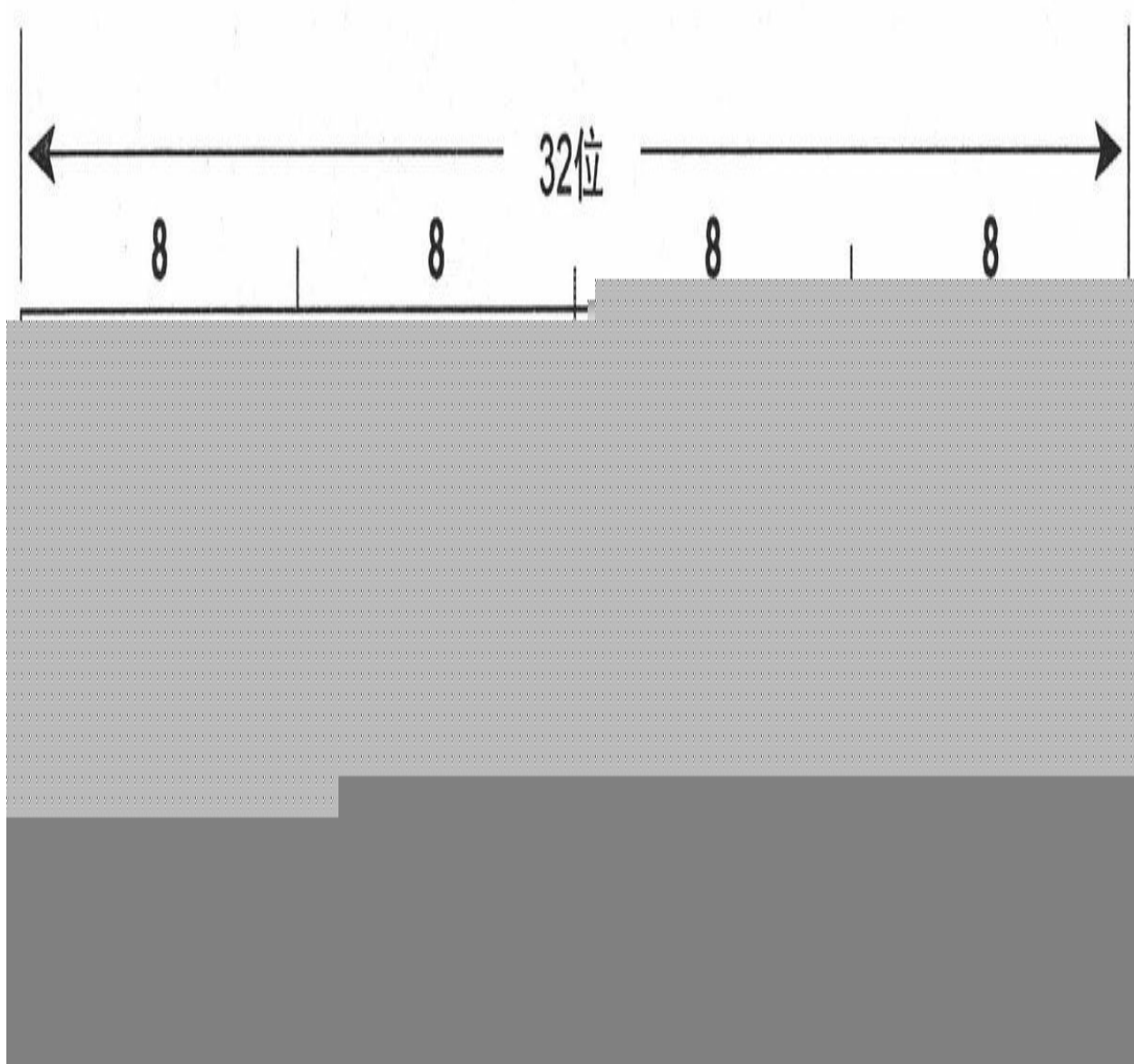
示例3-60 图3-16中**RTA**的路由表



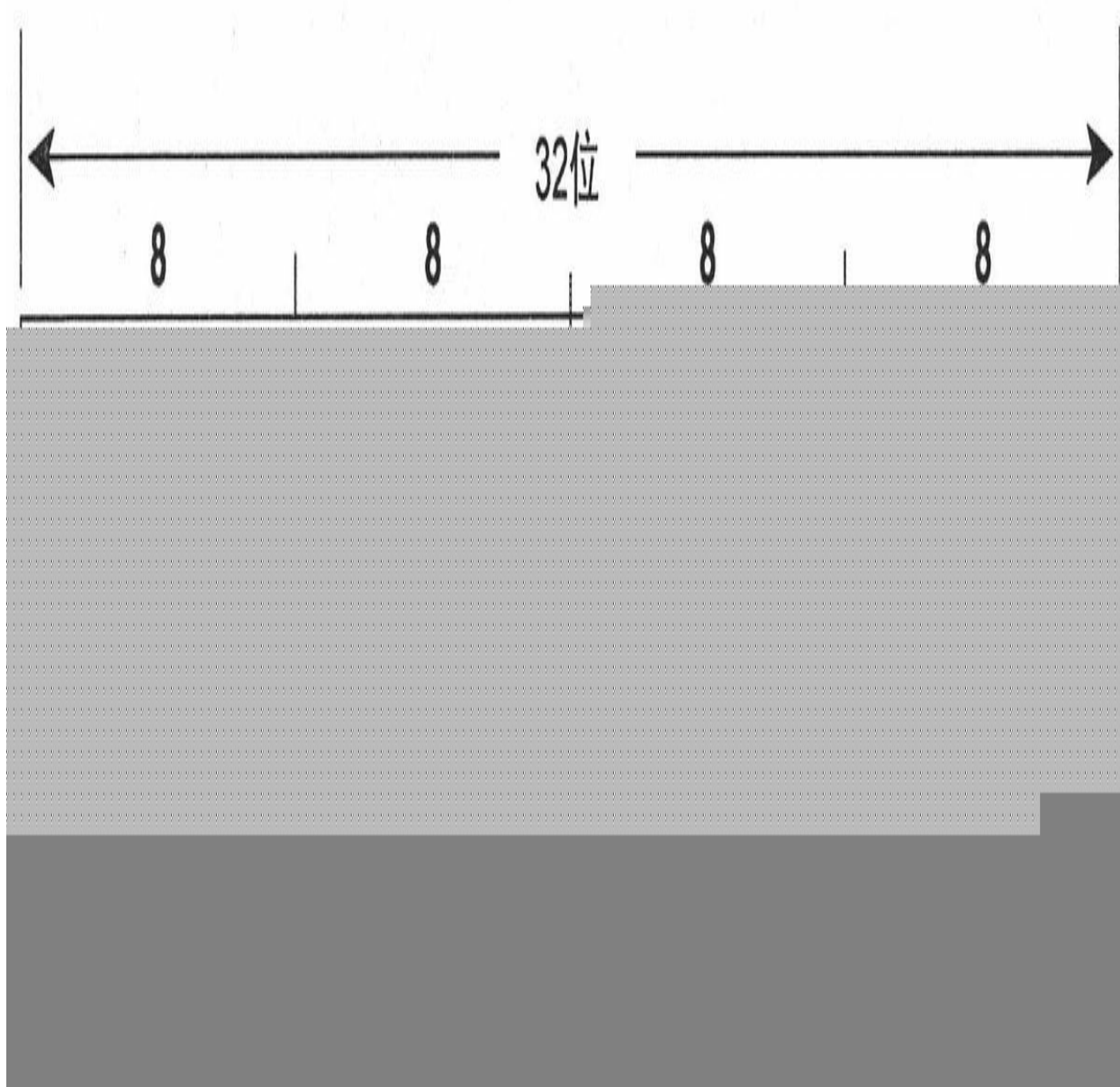
示例3-61 图3-16中RTB的路由表



示例3-62 图3-16中RTC的路由表



示例3-63 图3-16中RTD的路由表



[1] 还有一种特殊情况，那就是组播地址，它表示一组设备而不是所有设备。D类地址224.0.0.5就是一个组播地址，这个地址为所有OSPF路由器保留。

[2] 寻找最优匹配有两个基本过程，它们依赖于路由器是否表现为有类别或无类别。有类别路由表的查找过程参见第5章的相关内容，无类别路由表的查找参见第6章的相关内容。

[3] 要使本例及本章后续例子中的静态路由正常工作，必须附加两个全局

配置命令：`ip classless`和`ip subnet-zero`，在IOS 11.3及更高版本中，`ip classless`功能在缺省状态下是打开的。提及这两个命令主要考虑有读者希望在实验室中尝试这个配置实例，第6章将介绍这些命令。

[4] 为了表达清楚，删除了路由器上方的关键字。

[5] 接口上配置的地址是EUI-64格式的地址，因而对路由器来说，这些地址结合MAC地址是惟一的。

[6] 用小于特定类别地址标准掩码长度的掩码汇总一组主网地址的方法叫做超网化，详见第6章。

[7] 除非用扩展ping工具将源地址设置为其他地址。

[8] 如果在实验室做此练习，请确保在所有6台路由器上都配置命令**`ip classless`**。

本章包括以下主题：

- 路由选择协议基础；
- 距离矢量路由选择协议；
- 链路状态路由选择协议；
- 内部和外部网关协议；
- 静态或动态路由选择。

第4章

动态路由选择协议

上一章解释了路由器为了正确地交换数据包到达各自的目的地所需要知道的信息，以及怎样手工向路由表输入这些信息。本章将讨论路由器如何发现这些信息，并且通过动态路由选择协议与其他路由器共享这些信息。路由选择协议（**routing protocol**）作为路由器之间进行相互交流的语言，用于实现可达性信息和网络状态的共享。

动态路由选择协议不仅执行路径决策和路由表更新功能，而且还要在最优路径不可用时决策下一条最优路径。动态路由选择相比静态路由选择而言最大的优势在于，动态路由选择能够缓解拓扑变化带来的影响。

显然，为了正确地通信，通信双方必须使用相同的语言。自从IP路由选择出现以来，共有8种主要的IP路由选择协议可供选择；[\[1\]](#)如果一台路由器使用RIP与另一台使用OSPF的路由器进行对话，那么它们将无法实现信息共享，因为它们使用的语言不相同。后继章节将会分析所有目前在用的IP路由选择协议，甚至涉及如何使路由器“能说两种语言”，但是首先有必要研究一下所有路由选择协议共同的一些特性和问题——IP或其他方面。

4.1 路由选择协议基础

所有路由选择协议都是围绕着一一种算法而构建的。通常，一种算法是一个逐步解决问题的过程。一种路由算法至少应指明以下内容：

- 向其他路由器传送网络可达性信息的过程。
- 从其他路由器接收可达性信息的过程。
- 基于现有可达性信息决策最优路由的过程以及在路由表中记录这些信息的过程。
- 响应、修正和通告网络中拓扑变化的过程。

对所有路由选择协议来说，几个共同的问题是路径决策、度量、收敛和负载均衡。

4.1.1 路径决策

在网络内的所有子网都必须连接到一台路由器上，无论什么情况下，只要路由器有接口连接到一个网络上，那么该接口必须具有一个属于该网络的地址^[2]。这个地址就是可达性信息的起始点。

图4-1给出了一个包含3台路由器的网络。路由器A知道网络192.168.1.0、192.168.2.0和192.168.3.0的存在，因为路由器有接口连接到这些网络上，并且配置了相应的地址和掩码。同样的，路由器B知道网络192.168.3.0、192.168.4.0、192.168.5.0和192.168.6.0的存在，路由器C知道网络192.168.6.0、192.168.7.0和192.168.1.0的存在。由于每个接口都实现了所连接网络的数据链路和物理层协议，因此路由器也知道网络的状态（工作正常“up”或发生故障“down”）。

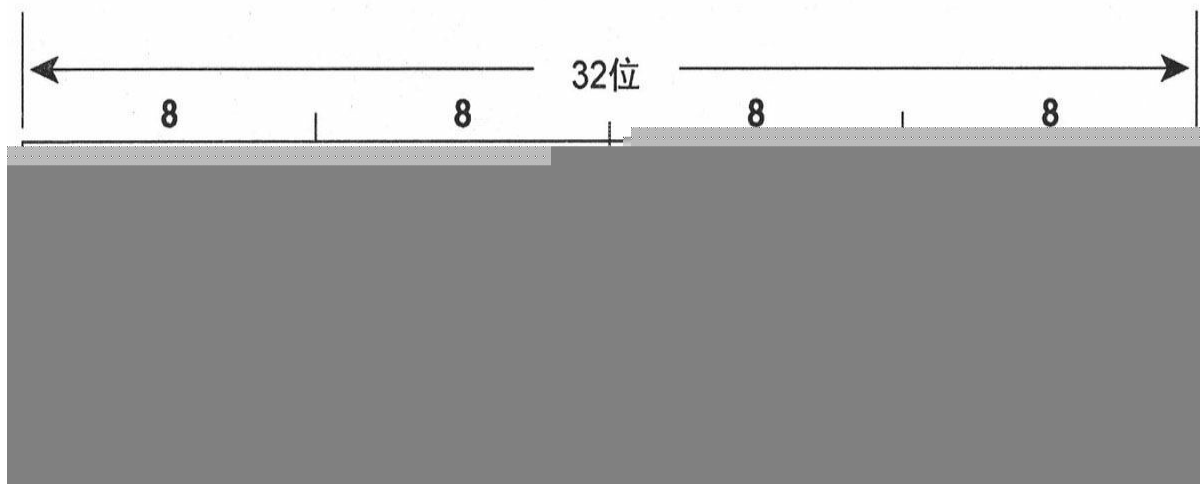


图4-1 每台路由器从分配给它的接口地址和掩码可以知道它的直连网络

信息共享过程看上去很简单。考虑路由器A：

步骤1： 路由器A检查自己的IP地址和相关掩码，然后推导出与自身所连接的网络是192.168.1.0、192.168.2.0和192.168.3.0。

步骤2： 路由器A将这些网络连同某种标记一起保存到路由表中，其中标记指明了网络是直连网络。

步骤3： 路由器A向数据包中加入以下信息：“我的直连网络是192.168.1.0、192.168.2.0和192.168.3.0。”

步骤4： 路由器A向路由器B和路由器C发送这些路由信息数据包的拷贝，或者叫做路由选择更新。

路由器B和路由器C执行与路由器A完全相同的步骤，并且也向路由器A发送带有与它们直接相连的网络的更新信息。路由器A将接收到的信息连同发送路由器的源地址一起写入路由表。现在，路由器A知道了所有的网络，而且还知道连接这些网络的路由器的地址。

这个过程看似非常简单。那么为什么路由选择协议比这更复杂呢？让我们重新看一下图4-1。

- 路由器A将来自路由器B和路由器C的更新信息保存到路由表之

后，它应该用这些信息作什么？例如，路由器A是否应该将路由器C的数据包信息传递给路由器B，还是将路由器B的路由选择信息包传递给路由器C呢？

- 如果路由器A没有转发这些更新消息，那么就不能完成信息共享。例如，如果路由器B和路由器C之间的链路不存在，那么这两台路由器就无法知道对方的网络。因此路由器A必须转发那些更新信息，但是这样做又产生了新的问题。
- 如果路由器A从路由器B和路由器C那里知道网络192.168.4.0，那么为了到达该网络应该使用哪一台路由器呢？它们都合法吗？谁是最优路径呢？
- 什么机制可以确保所有路由器能接收到所有的路由选择信息，而且这种机制还可以阻止更新数据包在网络中无休止地循环下去呢？
- 如果路由器共享某个直连网络（192.168.1.0、192.168.3.0和192.168.6.0），那么路由器是否仍旧应该通告这些网络呢？

这些问题同开头解释路由选择协议一样显得有点过分单纯，但是它们给读者的感觉却是：正是这些问题造成了协议的复杂性。每种路由选择协议无论如何都需要解决这些问题，这在下面的章节中将会变得更加清楚。

4.1.2 度量

当有多条路径到达相同目标网络时，路由器需要一种机制来计算最优路径。度量（metric）是指派给路由的一种变量，作为一种手段，度量可以按最好到最坏，或按最先选择到最后选择的顺序对路由进行等级划分。考虑下面的例子，了解为什么需要度量。

如图4-1所示，假设在网络中信息共享可以正常进行，并且路由器A中的路由表如表4-1所示。

表4-1 有关图4-1中路由器A的一个不完善的路由表

网络	下一跳路由器
----	--------

192.168.1.0	直接被连接
192.168.2.0	直接被连接
192.168.3.0	直接被连接
192.168.4.0	B, C
192.168.5.0	B, C
192.168.6.0	B, C
192.168.7.0	B, C

路由表说明了前3个网络直接连接到路由器，因而从路由器A到达它们不需要进行路由选择。根据路由表，后4个网络需要经过路由器B或路由器C才能到达。这些信息同样是正确的。但是，如果通过路由器B或路由器C都可以到达网络192.168.7.0，那么优先选择哪一条路径呢？这时就需要度量对这两条路径进行等级划分。

不同的路由选择协议使用不同的度量。例如，RIP定义含有路由器跳数最少的路径是最优路径；EIGRP基于路径沿路最小带宽和总时延定义最优路径。下一小节将给出这些度量和其他常用度量的基本定义。更复杂的内容——例如某些路由选择协议（EIGRP）怎样使用多个参数来计算度量以及如何处理度量值相同的路由——将在本书后面讲述各协议的章节中讨论。

1. 跳数

跳数（Hop Count）度量可以简单地记录路由器跳数。例如，如果数据包从路由器A的接口192.168.3.1发出，经过路由器B到达网络192.168.5.0，则记为1跳；如果从路由器接口192.168.1.1发出，经路由器C和路由器B到达网络192.168.5.0，记为2跳。假设仅使用跳数作为度量，那么最优路径就是跳数最少的路径，在本例中就是A-B。

但A-B是真正的最优路径吗？如果A-B是一条DS-0链路，并且A-C和C-B都是T1链路，那么跳数为2的路由实际上可能是最优路径，因为带宽对如何有效地使流量通过网络影响很大。

2. 带宽

带宽（Bandwidth）度量将会选择高带宽路径，而不是低带宽链路。然

而带宽本身可能不是一个好的度量。如果两条T1链路或其中一条被其他流量过多占用，那么与一个56KB的空闲链路相比到底谁好呢？或者一条高带宽但时延也很大的链路又如何呢？

3. 负载

负载（Load）度量反应了流量占用沿途链路带宽的数量。最优路径应该是负载最低的路径。

不像跳数和带宽，路径上的负载会发生变化，因而度量也会跟着变化。这里要当心，如果度量变化过于频繁，路由波动——最优路径频繁变化——可能就发生了。路由波动会对路由器的CPU、数据链路的带宽和全网稳定性产生负面影响。

4. 时延

时延（Delay）是度量数据包经过一条路径所花费的时间。使用时延作度量的路由选择协议将会选择最低时延的路径作为最优路径。有多种方法可以测量时延。时延不仅要考虑链路时延，而且还要考虑路由器的处理时延和队列时延等因素。另一方面，路由的时延可能根本无法测量。因此，时延可能是沿路径各接口所定义的静态延时量的总和，其中每个独立的时延量都是基于连接接口的链路类型估算得到的。

5. 可靠性

可靠性（Reliability）度量是用来测量链路在某种情况下发生故障的可能性，可靠性可以是变化的或固定的。可变可靠性度量的例子是链路发生故障的次数，或特定时间间隔内收到错误的次数。固定可靠性度量是基于管理员确定的一条链路的已知量。可靠性最高的路径将会被最优先选择。

6. 代价

由管理员设置的代价（Cost）度量可以反应更优或更差路由。通过任何策略或链路特性都可以对代价进行定义，同时代价也可以反应出网络管理员对路径的随意判断。因而代价是一个描述无量纲度量的术语。

每当谈论起路由选择的话题时，常常会把代价作为一种通用术语。例如，“RIP基于跳数选择代价最低的路径”。还有一个通用术语是最短（shortest），如“RIP基于跳数选择最短路径”。当在这种情况下使用它们时，最小代价（最高代价）或最短（最长）仅仅指的是路由选择协议基于自己特定的度量对路径的一种看法。

4.1.3 收敛

动态路由选择协议必须包含一系列过程，这些过程用于路由器向其他路由器通告本地的直连网络，接收并处理来自其他路由器的同类信息，以及传递从其他路由器接收到的信息。此外，路由选择协议还需要定义已确定的最优路径的度量。

对路由选择协议来说，另一个标准是网络上所有路由器的路由表中的可达性信息必须一致。在图4-1中，如果路由器A确定了经过路由器C到达网络192.168.5.0是最优路径，而路由器C确定到达相同网络的最优路径是经过路由器A，那么路由器A发向192.168.5.0的数据包到达路由器C后又被发回给路由器A，路由器A又再次发给路由器C，如此往复循环。我们称这种在两个或多个目标网络之间流量的持续循环为路由选择环路（routing loop）。

使所有路由表都达到一致状态的过程叫做收敛（convergence）。全网实现信息共享以及所有路由器计算最优路径所花费的时间总和就是收敛时间。

图4-2所示的网络已经收敛，但是现在拓扑又发生了变化。最左边的两台路由器之间的链路发生故障，这两台直接相连的路由器都从数据链路协议获知链路故障，转而通知它们的邻居该链路不再可用。邻接路由器立即更新路由表并通知它们的邻居，这个过程一直持续到所有路由器都知道此变化为止。

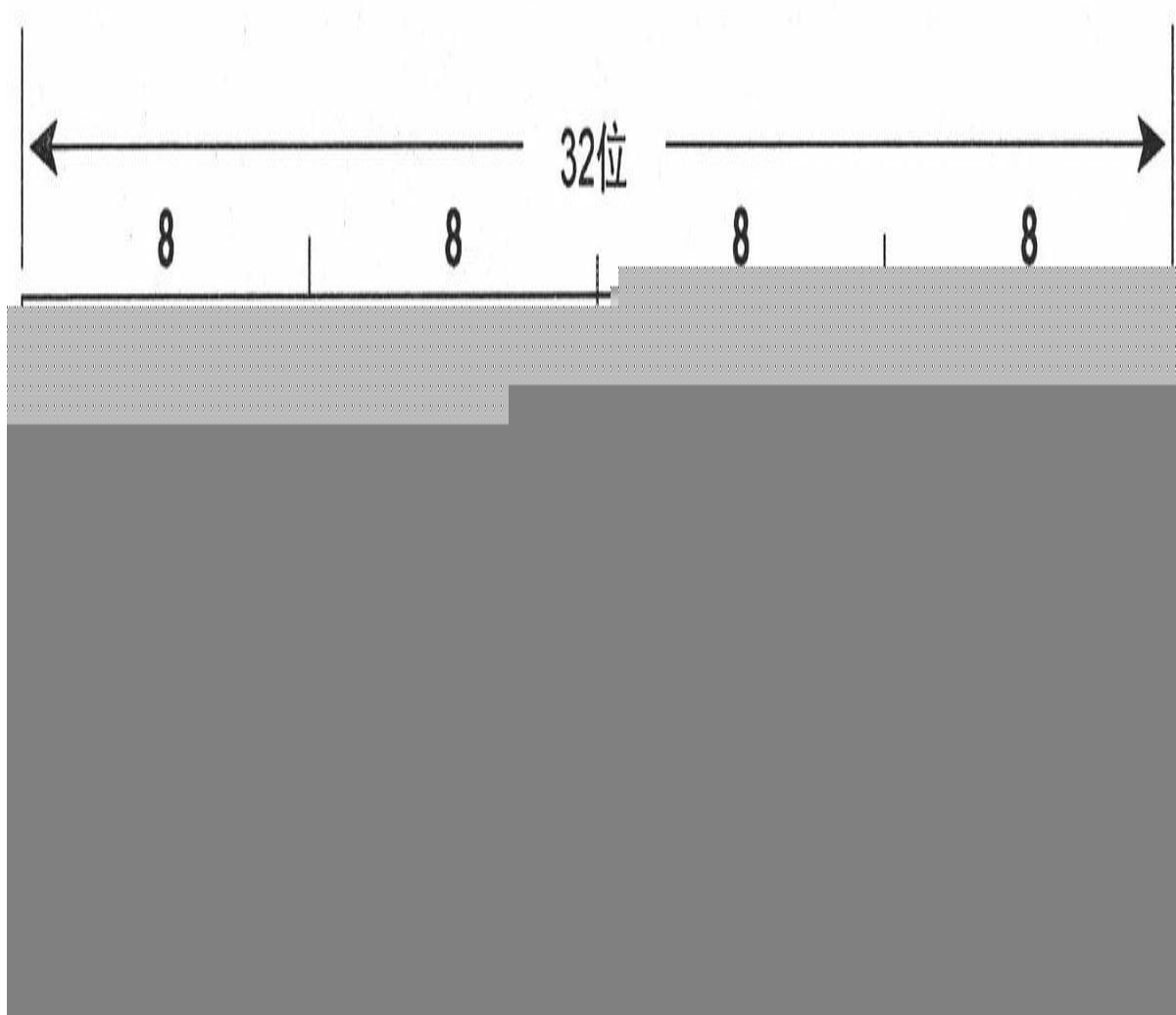


图4-2 拓扑发生变化后重新收敛需要一定时间。当网络处于未收敛状态时，路由器易受到错误路由选择信息的影响

注意，在 t_2 时刻，最左边的3台路由器知道拓扑发生了变化，但最右边的3台路由器依然不知道。最右边的3台路由器仍旧保存着原来的路由信息并继续交换数据包。这时网络处于未收敛状态，正是在这段时间里可能发生路由选择错误。因此，在任何路由选择协议里收敛时间是一个重要的因素。在拓扑发生变化之后，一个网络的收敛速度越快越好。

4.1.4 负载均衡

回忆一下第3章的内容，为了有效地使用带宽，负载均衡作为一种手段，将流量分配到相同目标网络的多条路径上。如图4-1所示，再让我

们考虑一下这个讨论负载均衡有效性的例子。图中所有网络都存在两条可达路径。如果网络192.168.2.0上的设备向192.168.6.0上的设备发送一组数据包流，路由器A可以经过路由器B或路由器C发送这些数据包。在这两种情况下，到目的网络的距离都是1跳。然而，在一条路径上发送所有的数据包不能最有效地利用可用带宽；因此应该执行负载均衡交替使用两条路径。正如第3章所述，负载均衡可以是等代价或不等代价，基于数据包或基于目标地址的。

4.2 距离矢量路由选择协议

大多数路由选择协议都属于如下两类的其中之一：距离矢量（distance vector）和链路状态（link state）。这里首先对距离矢量路由选择协议的基础内容进行分析，在下一节中将讨论链路状态路由选择协议。大多数距离矢量算法是以R.E.Bellman^[3]、L.R.Ford和D.R.Fulkerson^[4]所做的工作为基础的，由于这个原因，所以有时距离矢量算法又称为*Bellman-Ford* 或*Ford-Fulkerson* 算法。但值得注意的是EIGRP是一个例外情况，它是基于J.J.Garcia Luna Aceves开发的算法实现的。

距离矢量名称的由来是因为路由是以矢量（距离，方向）的方式被通告出去的，其中距离是根据度量定义的，方向是根据下一跳路由器定义的。例如，“目标A在下一跳路由器X的方向，距此5跳之远”。这个表述隐含了每台路由器向邻接路由器学习它们所观察到的路由信息，然后再向外通告自己观察到的路由信息。因为每台路由器在信息上都依赖于邻接路由器，而邻接路由器又从它们的邻接路由器哪里学习路由，依次类推，所以距离矢量路由选择有时又被认为是“依照传闻进行路由选择”。

下面都属于距离矢量路由选择协议：

- IP路由选择信息协议（RIP）；
- Xerox网络系统的XNS RIP；
- Novell的IPX RIP；
- Cisco Systems的Internet网关路由选择协议（IGRP）和增强型Internet网关路由协议（EIGRP）；
- DEC的DNA阶段4；
- Apple Talk的路由选择表维护协议（RTMP）。

4.2.1 通用属性

典型的距离矢量路由选择协议通常会使用一个路由选择算法，算法中路

由器通过广播整个路由表，定期地向所有邻居发送路由更新信息。[\[5\]](#)

上面这个表述包含了大量信息，下面将更加详细地讨论这些内容。

1. 定期更新（**Periodic Updates**）

定期更新意味着每经过特定时间周期就要发送更新信息。这个时间周期从10s（AppleTalk的RTMP）到90s（Cisco的IGRP）。这里有争议的是如果更新信息发送过于频繁可能会引起拥塞；但如果更新信息发送不频繁，收敛时间可能长的不能被接收。

2. 邻居（**Neighbours**）

在路由器上下文中，邻居通常意味着共享相同数据链路的路由器或某种更高层的逻辑邻接关系。距离矢量路由选择协议向邻接路由器[\[6\]](#)发送更新信息，并依靠邻居再向它的邻居传递更新信息。因此，距离矢量路由选择被说成使用逐跳更新方式。

3. 广播更新（**Broadcast Updates**）

当路由器首次在网络上被激活时，路由器怎样寻找其他路由器呢？它将如何宣布自己的存在呢？这里有几种方法可以采用。最简单的方法是向广播地址发送（在IP网络中，广播地址是255.255.255.255）更新信息。使用相同路由选择协议的邻居路由器将会收到广播数据包并且采取相应的动作。不关心路由更新信息的主机和其他设备仅仅丢弃该数据包。

4. 全路由选择表更新

大多数距离矢量路由选择协议使用非常简单的方式告诉邻居它所知道的一切，该方式就是广播它的整个路由表，但下面我们会讨论几个特例。邻居在收到这些更新信息之后，它们会收集自己需要的信息，而丢弃其他信息。

[4.2.2 依照传闻进行路由选择](#)

在图4-3中，正在执行一个距离矢量算法，其中使用跳数作为度量。在 t_0 时刻，路由器A到路由器D正好可用。让我们沿最上面一行查看路由表，在 t_0 时刻4台路由器所具有的惟一信息就是它们的直连网络。路由表标识了这些网络，并且指明它们没有经过下一跳路由器，是直接连接到路由器上的，所以跳数为0。每台路由器都将在它所有的链路上广播这些信息。

在 t_1 时刻，路由器接收并处理第1个更新信息。查看此时路由器A的路由表，路由器B发给路由器A的更新信息发现路由器B能够到达网络10.1.2.0和10.1.3.0，而且距离都为0跳。如果这些目标网络距离路由器B为0跳，那么距离路由器A则为1跳。所以路由器A将跳数增加1，然后检查自己的路由表。路由表中显示网络10.1.2.0已知，且距离为0跳，小于路由器B通告的跳数，所以路由器A忽略此信息。

由于网络10.1.3.0对路由器A来说是新信息，所以路由器A将其输入到路由表中。更新数据包的源地址是路由器B的接口地址（10.1.2.2），因此该地址连同计算的跳数一起也被保存到路由表中。

注意，在 t_1 时刻其他路由器也执行了类似的操作。例如，路由器C忽略了来自路由器B关于10.1.3.0的信息以及来自路由器C关于10.1.4.0的信息，但是保存了以下信息：经过路由器B的接口地址10.1.3.1可以到达网络10.1.2.0以及经过路由器C的接口地址10.1.4.2可以到达网络10.1.5.0。经计算得知路由器C到达这两个网络的距离都为1跳。

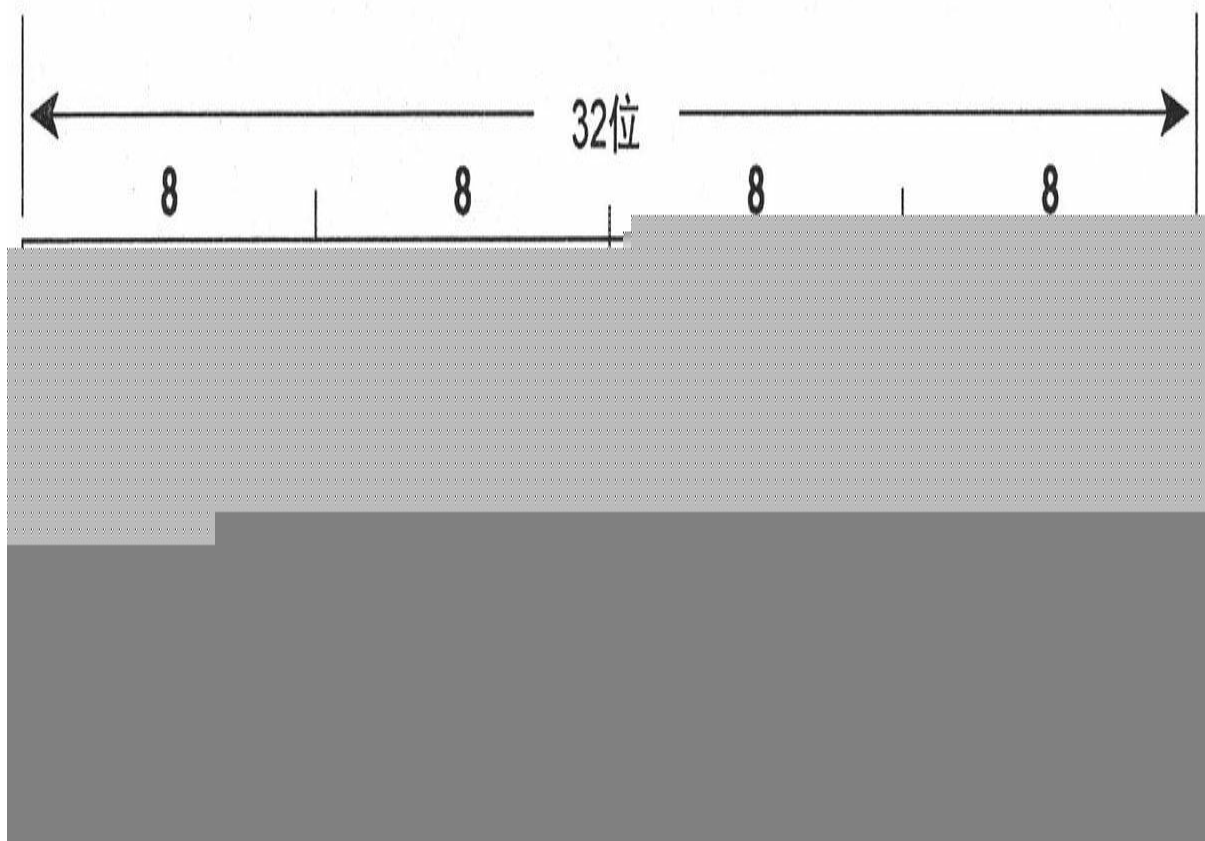


图4-3 距离矢量协议逐跳收敛

在 t_2 时刻，随着更新周期再次到期，另一组更新消息被广播。路由器B发送了最新的路由表；路由器A再次将路由器B通告的跳数加1后与自己的路由表比较。像上次一样，路由器A又一次丢弃了关于10.1.2.0的信息。由于网络10.1.3.0已知且跳数没有发生变化，所以该信息也被丢弃。惟有10.1.4.0被作为新的信息输入到路由表中。

在 t_3 时刻，网络已收敛。每台路由器都已经知道了每个网络以及到达每个网络的下一跳路由器的地址和距离跳数。

这里打个比方。你正在新墨西哥洲北部的Sangre de Cristo山中漫步，如果你不会迷路的话，这里是一个迷人的漫步场所。但是如果你迷路了且遇到一个叉路口，一个路标指向西面，上面写着“陶斯镇，15英里”。这时你除了相信这个路标外别无选择。你不知道15英里外的地形是什么，你也不知道是否有更好的路，或者这个路标是否正确。如果有人将路标转个方向，那么你不但不能去往安全的地方反而走向森林的更深处！

距离矢量算法提供了指向网络的路标。[\[7\]](#)该算法给出了方向和距离，但是没有给出沿着这条路径行走的细节。就像叉路口的路标一样，它很容易受到意外或故意的误导。下面是距离矢量算法所面临的一些困难及算法的改进。

[4.2.3 路由失效计时器](#)

在图4-3中，网络已经收敛，那么当部分网络拓扑发生变化时，它怎样处理重新收敛问题呢？如果网络10.1.5.0发生故障，答案很简单——在下一个更新周期中，路由器D将这个网络标记为不可达并且发送该信息。

但是如果网络10.1.5.0没有故障，而是路由器D发生故障该怎么办？这时路由器A、路由器B和路由器C的路由表中仍然保存着关于网络10.1.5.0的信息，虽然该信息不再有用，但是却没有路由器通知它们。它们将不知不觉地向一个不可达的网络转发着数据包——即在网络中打开了一个黑洞。

处理这个问题的办法是为路由表中的每个表项设置路由失效计时器。例如，当路由器C首次知道10.1.5.0并将其输入到路由表中时，路由器C将为该路由设置计时器。每隔一定时间间隔路由器C都会收到路由器D的更新信息，路由器C在丢弃有关10.1.5.0的信息的同时复位该路由的计时器。

如果路由器D发生故障，路由器C将不能接收到关于10.1.5.0的更新信息。这时计时器将会超时，路由器C将该路由标记为不可达，并将在下一个更新周期时传递该信息。

路由超时的典型周期范围是3~6个更新周期。路由器在丢失单个更新信息之后将不会使路由无效的，因为数据包的损坏、丢失或者某种网络延时都会造成这种事件的发生。但是，如果路由失效周期太长，网络收敛速度将会过慢。

[4.2.4 水平分隔](#)

根据到目前为止所描述的距离矢量算法，每台路由器在每个更新周期都要向每个邻居发送它的整个路由表。但是这真的有必要吗？在图4-3

中，路由器A知道的每个距离大于0跳的网络都是从路由器B学习来的。常识表明，如果路由器A将学自路由器B的网络再广播给路由器B，那么这是一种资源浪费。显然，路由器B已经知道这些网络。

路由的指向与数据包流动方向相反的路由被称为逆向路由（reverse route）。水平分隔（split horizon）是一种在两台路由器之间阻止逆向路由的技术。

这样做除了不会浪费资源，还有一个很重要的原因是不会把从路由器学习到的可达性信息再返回给这台路由器。动态路由选择协议最重要的功能是监测和补偿拓扑变化——如果到网络的最优路径不可用，协议必须寻找下一个最优路径。

再看一下图4-3中已收敛的网络，假设网络10.1.5.0发生故障。路由器D监测到该故障，将网络标记为不可达并在下一更新周期通知路由器C。然而在路由器D的更新计时器触发更新之前，意想不到的事情发生了。路由器C的更新消息到达了路由器D，声明路由器C可以到达网络10.1.5.0，距离为1跳！还记得上面路标的比喻吗？路由器D不知道路由器C通告的下一条最优路径并不合理，因而路由器D将跳数加1并在路由表中记录以下信息：通过路由器C的接口（10.1.4.1）可以到达网络10.1.5.0，距离为2跳。

此时目标地址为10.1.5.3的数据包到达路由器C，路由器C查询路由表并将数据包转发给路由器D。路由器D查询路由表又将数据包转发给路由器C，路由器C再转回给路由器D，一直无穷尽地进行下去，因而导致路由环路的发生。

执行水平分隔可以阻止路由环路的发生。有两类水平分隔方法：简单水平分隔法和毒性逆转水平分隔法。

简单水平分隔的规则是，从某接口发送的更新消息不能包含从该接口收到的更新所包含的网络。

在图4-4中，路由器执行简单的水平分隔。路由器C向路由器D发送了关于网络10.1.1.0、10.1.2.0和10.1.3.0的更新信息。其中没有包含网络10.1.4.0和10.1.5.0，因为它们是从路由器D获取的。同样的，发送给路由器B的更新信息包括了网络10.1.4.0和10.1.5.0，而没有提及10.1.1.0、10.1.2.0和10.1.3.0。

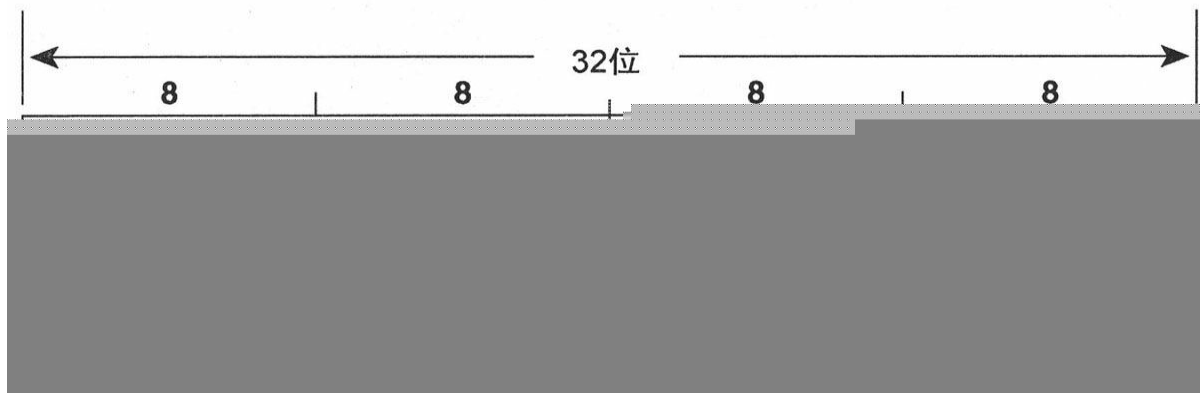


图4-4 简单水平分隔没有把从邻居那里获取的路由通告给邻居

简单水平分隔采用抑制信息的工作方式。毒性逆转水平分隔法是一种改进方法，它可以提供更积极的信息。

毒性逆转水平分隔法的规则是，当更新信息被发送出某接口时，信息中将指定从该接口收到的更新信息中获取的网络是不可达的。

在图4-4中，事实上，路由器C向路由器D通告了网络10.1.4.0和10.1.5.0，但是这些网络都被标记为不可达。图4-5给出了从路由器C到路由器B和路由器D的路由表，看上去很相似。注意，通过设置度量为无穷大可以标记网络为不可达；换言之，网络无穷远。下一小节将讨论路由选择协议中无穷大的概念。

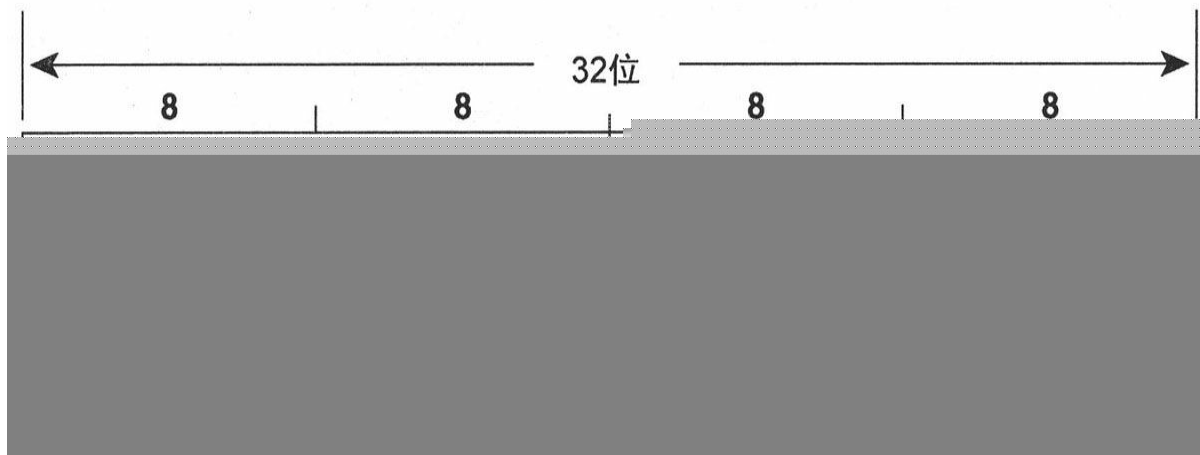


图4-5 虽然毒性逆转水平分隔法通告逆向路由，但使用了不可达度量（无穷大）

毒性逆转水平分隔法被认为比简单水平分隔法更安全更健壮——一种“坏信息总比没消息好”的方法。例如在图4-5中，假设路由器B收到错误信息使其相信经过路由器C可以到达子网10.1.1.0。简单水平分隔法无法纠正这种错误，而路由器C的毒性逆转更新信息可以立刻制止这种潜在的环路。正因如此，大部分现代距离矢量算法的实现都使用了毒性逆转水平分隔法。缺点是使路由更新数据包更大了，可能会加剧链路的拥塞问题。

4.2.5 计数到无穷大

水平分隔法切断了邻居路由器之间的环路，但是它不能割断网络中的环路，如图4-6所示，这里还是10.1.5.0发生故障。路由器D向路由器C（虚箭头）和路由器B（实箭头）发送了相应的更新信息。于是路由器B将经过路由器D的路由标记为不可达，而此时路由器A正在向外通告到达10.1.5.0的次最优路径，距离为3跳；因此路由器B在路由表中记录下此路由。

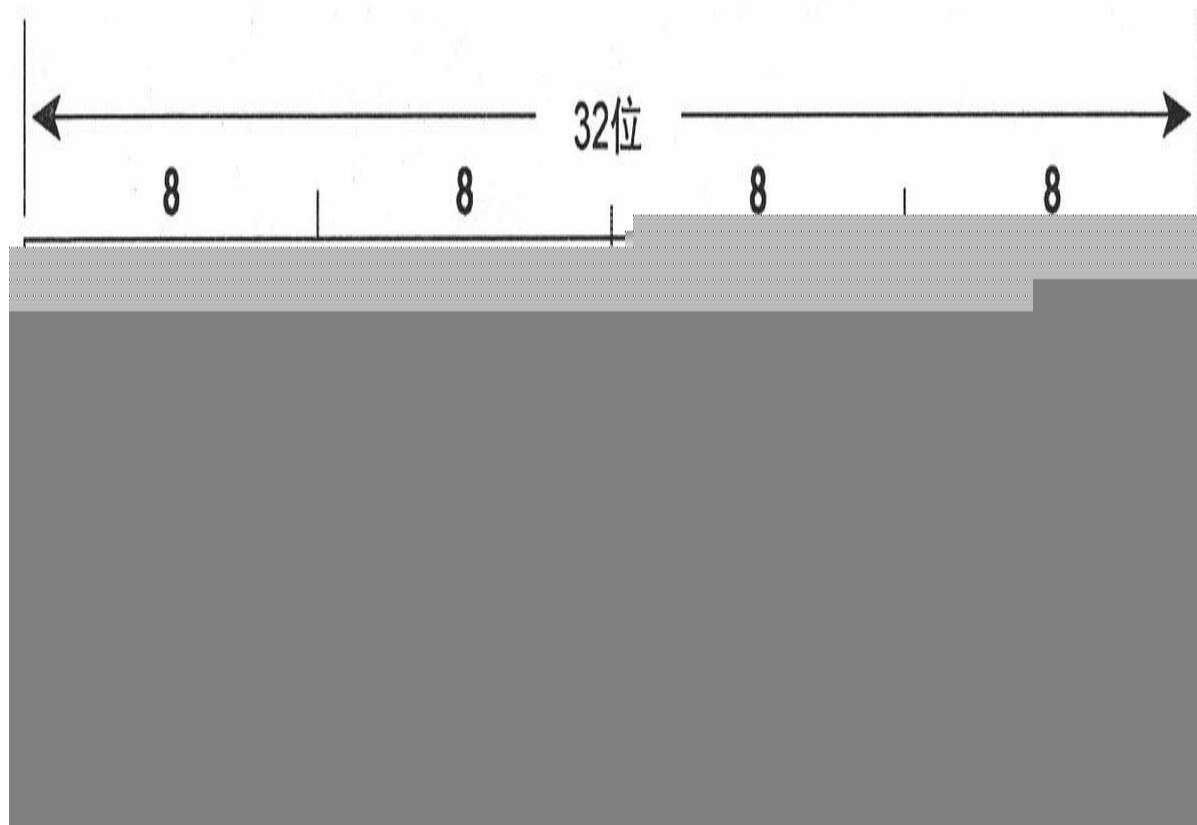


图4-6 水平分隔法不能阻止这里的路由环路

路由器B现在又通知路由器D它有另一条路由可以到达10.1.5.0。于是路由器D也记录下这个路由，并通知路由器C它有一条距离10.1.5.0为4跳的路由。路由器C又告诉路由器A距离10.1.5.0有5跳远。路由器A告诉路由器B有6跳远。

“啊”，路由器B想，“虽然路由器A到网络10.1.5.0的路径在不断加长。不过它是惟一可用的路径，所以我将使用这条路径！”

路由器B又将跳数加到7，并通知路由器D，如此循环下去。这种情况就叫计数到无穷大，因为到10.1.5.0的跳数将会持续增加到无穷大。虽然所有路由器都执行了水平分隔，但对此无能为力。

减轻计数到无穷大影响的方法是定义无穷大。大多数距离矢量协议定义无穷大为16跳。在图4-6中，随着更新消息在路由器中转圈，到10.1.5.0的跳数最终将增加到16。那时网络10.1.5.0将被认为不可达。

这也是路由器如何通告网络不可达的一种方法。一个网络发生故障，不管它是毒性逆转路由，还是超过最大网络尺寸15跳的路由，路由器将把所有跳数为16的路由看作不可达。

设置最大跳数15有助于解决计数到无穷大的问题，但是收敛速度仍旧非常慢。假设更新周期为30s，网络可能花7.5min达到收敛，在这期间容易受到路由错误的影响。触发更新可以用于减少收敛时间。

4.2.6 触发更新

触发更新（Triggered Update）又叫快速更新，非常简单：如果一个度量变好或变坏，那么路由器将立即发送更新信息，而不等更新计时器超时。这样重新收敛的速度将会比每台路由器必须等待更新周期的方式快，而且可以大大减少计数到无穷大所引发的问题，虽然不能完全消除。定期更新和触发更新可能会一起发生，因而路由器可能会在收到来自触发更新的正确信息之后又收到来自未收敛路由器的错误信息。这种情况表明，当网络正在进行重新收敛时，还会发生混乱和路由错误。但是触发更新将有助于更快地消除这些问题。

对触发更新进一步的改进是更新信息中仅包括实际触发该事件的网络，而不是包括整个路由表。触发更新技术减少了处理时间和对网络带宽的影响。

4.2.7 抑制计时器

触发更新为正在重新进行收敛的网络增加了应变能力。为了降低接受错误路由选择信息的可能性，抑制计时器（Holddown Timer）引入了某种程度的怀疑量。

如果到一个目标的距离增加（例如，跳数由2增加到4），那么路由器将为该路由设置抑制计时器。直到计时器超时，路由器才可以接受有关此路由的更新信息。

显然，这也是一种折衷办法。错误路由选择信息进入路由表的可能性被减小了，但是重新收敛的时间也被耗费了。像其他计时器一样，必须小心设置抑制计时器。如果挂起时间太短，不起作用；如果太长，正常路由则会受到不利的影响。

4.2.8 异步更新

图4-7给出了一组连接在以太网骨干上的路由器。路由器将不会同时广播更新信息，如果发生这种情况，更新数据包会发生碰撞。但是当几台路由器共享一个广播网络时可能会发生这种情况。因为在路由器中，更新处理所带来的系统时延将导致更新计时器趋于同步。当几台路由器的计时器同步后，碰撞随之发生，这又进一步影响到系统时延，最终共享广播网络的所有路由器都可能变得同步起来。

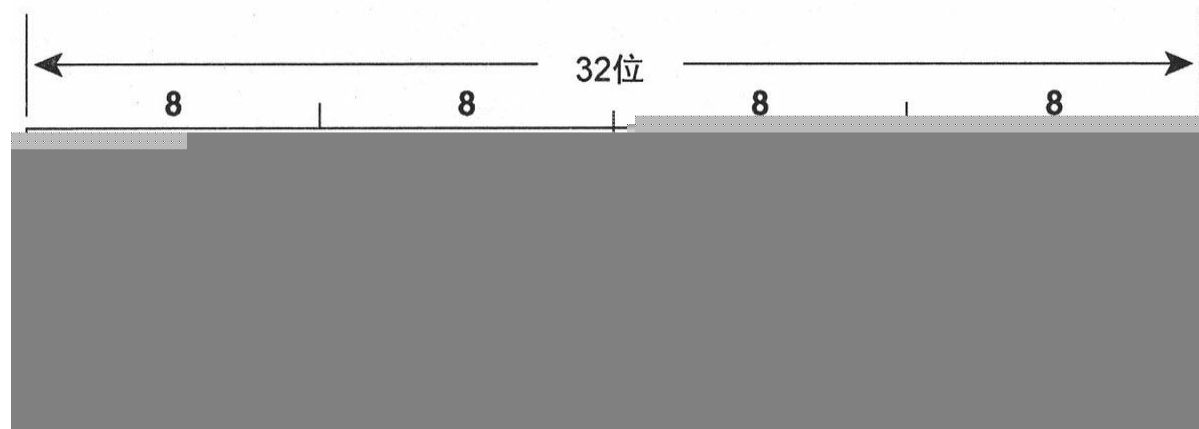


图4-7 如果更新计时器同步，就可能发生碰撞

可以使用下面两种方法维持异步更新（Asynchronous Update）：

- 每台路由器的更新计时器独立于路由选择进程，因而不会受到路由器处理负载的影响。
- 在每个更新周期中加入一个小的随机时间或定时抖动作为偏移。

如果路由器实现了严格的、独立于系统的计时器方法，那么还必须按照随机的方式激活共享一个广播网络的所有路由器。因为并行启动整组路由器——比如发生大面积断电——可能造成所有计时器同时发送更新信息。

如果随机变量与共享广播网络的路由器数量相比足够大，那么增加更新周期的随机性是有效的。Sally Floyd和Van Jacobson^[8]曾作过计算，在足够大的网络中，过小的随机化会被路由器所克服，为了保证有效性，更新计时器应该分布在中等更新周期的50%的范围内。

4.3 链路状态路由选择协议

距离矢量路由器所使用的信息可以比拟为由路标提供的信息。链路状态路由选择协议像是一张公路线路图。链路状态路由器是不容易被欺骗而作出错误的路由决策的，因为它有一张完整的网络图。链路状态不同于距离矢量依照传闻进行路由选择的工作方式，原因是链路状态路由器从对等路由器^[9]那里获取第一手信息。每台路由器会产生一些关于自己、本地直连链路、这些链路的状态（以此而得名）和所有直接相连邻居的信息。这些信息从一台路由器传送到另一台路由器，每台路由器都做一份信息拷贝，但是决不改动信息。最终目的是每台路由器都有一个相同的有关网络的信息，并且每台路由器可以独立地计算各自的最优路径。

链路状态协议，有时叫最短路径优先协议或分布式数据库协议，是围绕着图论中的一个著名算法——E.W.Dijkstra的最短路径算法设计的。链路状态协议有以下几种：

- IP开放式最短路径优先（OSPF）；
- CLNS或IPISO的中间系统到中间系统（IS-IS）；
- DEC的DNA阶段5；

- Novell的NetWare链路服务协议（NLSP）。

虽然链路状态协议确实考虑的比距离矢量协议更复杂，但是基本功能却一点也不复杂。

步骤1： 每台路由器与它的邻居之间建立联系，这种联系叫做邻接关系。

步骤2： 每台路由器向每个邻居[\[10\]](#)发送被称为链路状态通告（LSA）的数据单元。对每台路由器链路都会生成一个LSA，LSA用于标识这条链路、链路状态、路由器接口到链路的代价度量值以及链路所连接的所有邻居。每个邻居在收到通告之后将依次向它的邻居转发（泛洪）这些通告。

步骤3： 每台路由器要在数据库中保存一份它所收到的LSA的备份。如果所有工作正常，所有路由器的数据库应该相同。

步骤4： 完整的拓扑数据库，也叫链路状态库，Dijkstra算法使用它对网络图进行计算得出到每台路由器的最短路径。接着链路状态协议对链路状态数据库进行查询找到每台路由器所连接的子网，并把这些信息输入到路由表中。

[4.3.1 邻居](#)

邻居发现是建立链路状态环境并运转的第一步。与友邻术语一样，这一步将使用Hello 协议（Hello Protocol）。Hello协议定义了一个Hello数据包的格式和交换数据包并处理数据包信息的过程。

Hello数据包至少应包含一个路由器ID CRID和发送数据包的网络地址。路由器ID可以将发送该数据包的路由器与其他路由器惟一地区分开。例如，路由器ID可以是路由器的一个接口的IP地址。数据包的其他字段可以携带子网掩码、Hello间隔、线路类型描述符和帮助建立邻居关系的标记，其中Hello间隔是路由器在宣布邻居死亡之前等待的最大周期。

当两台路由器已经互相发现并将对方视为邻居时，它们要进行数据库同步过程，即交换和确认数据库信息直到数据库相同为止。数据库同步的细节参见第8章和第9章的内容。为了执行数据库同步，邻居之间必须建立邻接关系，即它们必须就某些特定的协议参数，如计时器和对可选择

能力的支持，达成一致意见。通过使用Hello数据包建立邻接关系，链路状态协议就可以在受控的方式下交换信息。与距离矢量相比，这种方式仅在配置了路由选择协议的接口上广播更新信息。

除建立邻接关系之外，Hello数据包还可作为监视邻接关系的握手信号。如果在特定的时间内没有从邻接路由器收到Hello数据包，那么认为邻居路由器不可达，随即邻接关系被解除。典型的Hello数据包交换间隔为10s，典型的死亡周期是数据包交换间隔的4倍。

4.3.2 链路状态泛洪扩散

在建立邻接关系之后，路由器开始发送LSA。从术语泛洪扩散（Flooding）我们可以得知，通告被发送给每个邻居。路由器保存接收到的LSA，并依次向每个邻居转发，除了发送该LSA的邻居之外。这个过程是链路状态优于距离矢量的一个原因。LSA几乎是立刻被转发，而距离矢量在发送路由更新之前必须运行算法并更新自身的路由表，甚至对触发路由更新也是如此。因此，当网络拓扑改变时，链路状态协议收敛速度远远快于距离矢量协议。

泛洪扩散过程是链路状态协议中最复杂的一部分。有几种方式可以使泛洪扩散更高效和更可靠，如使用单播和组播地址、校验和以及主动确认。在有关具体协议的章节中将讨论这些主题，但是有两个过程对泛洪扩散是极其重要的：排序和老化。

1. 序列号

到目前为止，泛洪扩散的一个难点是当所有路由器收到所有LSA时，泛洪扩散必须停止。我们可以希望数据包中的TTL值终止过期的数据包，但是之前几乎不可能有效地控制LSA在网络中漫游。如图4-8所示，路由器A连接的子网172.22.4.0发生故障，因而路由器A向邻居路由器B和路由器D泛洪扩散一个LSA，以便通告该链路的新状态。路由器B和路由器D忠实地向它们的邻居扩散该LSA，依次类推。

看一下在路由器C上会发生什么。在 t_1 时刻，一个来自路由器B的LSA到达路由器C，路由器C将相关信息输入到拓扑数据库，并且向路由器F进行转发。在 t_3 时刻，另一份相同的LSA拷贝经A-D-E-F-C路径到达路由器C。路由器C发现数据库中已经存在该LSA，问题是路由器C是否应该

向路由器B转发这个LSA?答案是不转发,因为路由器B已经收到了这个通告。由于路由器C从路由器F接收到的LSA的序列号与早先从路由器B接受的LSA的序列号相同,所以路由器C也知道这一情况。

当路由器A发送LSA时,在每个拷贝中的序列号都是一样的。这个序列号和LSA的其他部分一起被保存在路由器的拓扑数据库中,当路由器收到数据库中已存在的LSA且序列号相同时,路由器将丢弃这些信息。如果信息相同但是序列号更大,那么接收的信息和新序列号被保存到数据库中,并且泛洪扩散该LSA。按照这种方式,当所有路由器都收到LSA的最新拷贝时,泛洪扩散将停止。

到目前为止,根据所描述的,路由器好像仅对数据库中LSA与新收到的LSA是否相同进行验证,并据此作出扩散/丢弃决定,并没有使用序列号。但是假设图4-8中的网络172.22.4.0在发生故障后马上恢复正常。路由器A可能会发出通告网络故障的LSA,序列号为166,接着再发送通告网络正常的LSA,序列号为167。路由器C先后接收到沿路径A-B-C扩散过来的LSA,分别是关于网络发生故障和故障恢复的通告,但是接着路由器C又收到沿路径A-D-E-F-C扩散过来的关于网络故障的LSA。没有序列号,路由器C将不知道是否应该相信最后到达的关于网络故障的LSA。而使用序列号,路由器C的数据库可以显示来自路由器A的LSA的序列号为167,而后面到达的LSA序列号为166,因此该LSA被认为是过时的信息而被丢弃。

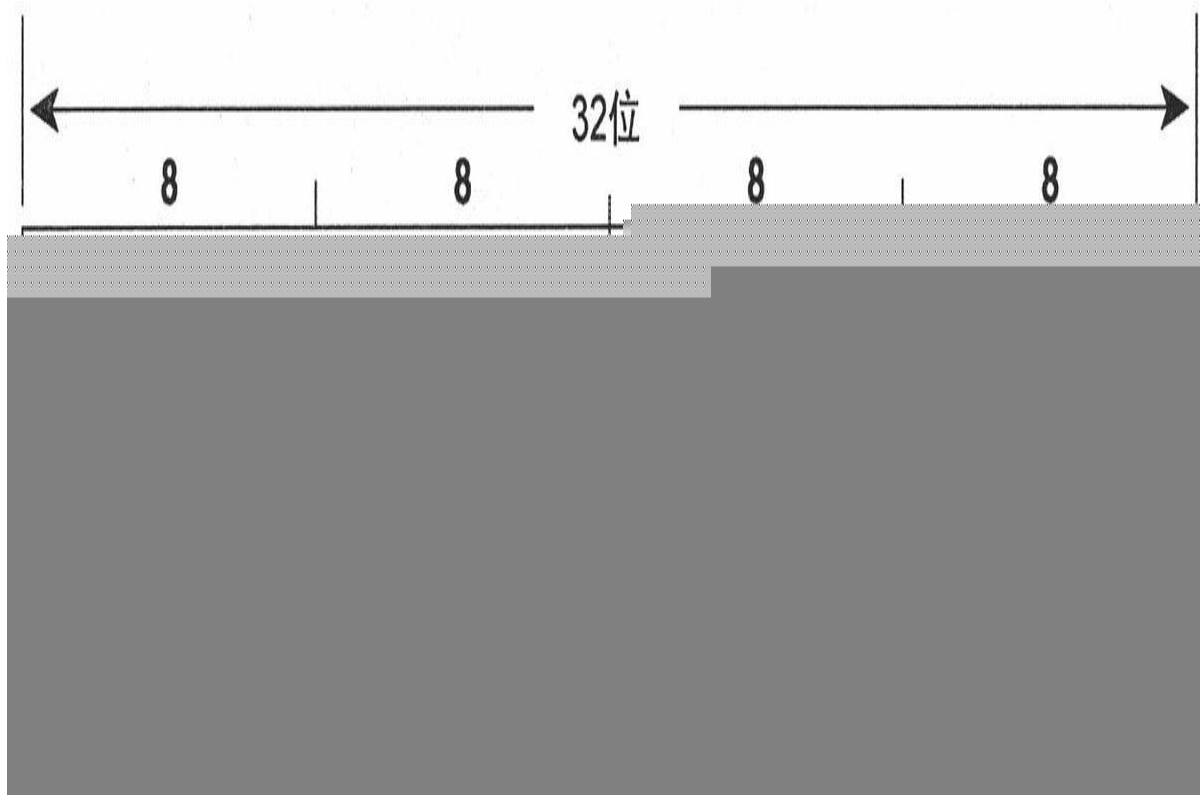


图4-8 当拓扑发生变化时，通告该变化的LSA在整个网络上扩散

因为序列号被携带在LSA中的一个固定字段内，所以序列号一定有上限，那么当序列号到达上限时会发生什么呢？

（1）线性序列号空间

一种办法是使用一个非常大的线性序列号空间以至于根本不可能到达上限。例如，如果使用32位长字段，那么从0开始将有 $2^{32} = 4\,294\,967\,296$ 个可用序列号。即使路由器每10s产生一个新的状态数据包，它也需要花1361年才能用尽所有序列号。几乎没有路由器被希望持续这样长的时间。

很不幸，在这个不够完美的世界上，故障依然会发生。如果一个链路状态路由选择进程用完了所有序列号，那么它在重新使用最低序列号之前必须停止，并等待它所发出的LSA在所有数据库中都不再被使用（参见本章“老化（Aging）”部分的内容）。

在路由器启动期间会出现一种更常见的困难。如果路由器A重启后，它

无法记得上次使用的序列号，必须重新使用1。但是如果路由器A的邻居仍然在数据库中保留了路由器A上次的序列号，那么越小的序列号也就是越旧的序列号，因而会被忽略。而且，路由选择进程必须一直等到网络上所有陈旧的LSA都消失为止。假如最大的时间是1个小时或更长，那么这种方法没有任何吸引力。

更好的解决办法是在泛洪扩散行为中添加新的规则：如果一台重新启动的路由器向邻居发送LSA的序列号比邻居保存的序列号还要老，那么邻居会发回自己保存的LSA和序列号。这台路由器将知道启动前自己使用的序列号并作出相应的调整。

然而需要小心，最近使用过的序列号不能接近上限。否则，重新启动的路由器将不得不再次重新启动。必须设定规则限制路由器“跳跃”地使用序列号。例如，规则指明一次序列号的增加不能超过整个序列号空间的二分之一（实际公式比例子要复杂，因为要考虑年龄的限制）。

IS-IS使用32位线性序列号空间。

（2）循环序列号空间

另一种方法是使用循环序列号空间，数字是循环使用的——在32位空间内紧跟在4 294 967 295后面的是0。然而在这里的故障也会让人左右为难，重新启动的路由器可能会遇到同线性序列号一样的问题。

循环序列编号建立了一个不合逻辑的奇特的位。如果 x 是1到4 294 967 295之间任意的一个数，那么 $0 < x < 0$ 。在运行正常的网络中通过声明两条规则可以维持这种条件，其中声明的规则用来确定什么时候一个序列号大于或小于另一个序列号。假设序列号空间为 n ，有两个序列号 a 和 b ，如果满足以下任意一种条件，则认为 a 更新（数量更大）：

- $a > b$ 且 $(a - b) \leq n/2$
- $a < b$ 且 $(b - a) > n/2$

为了简单起见，如图4-9所示，我们使用一个6位序列号空间：

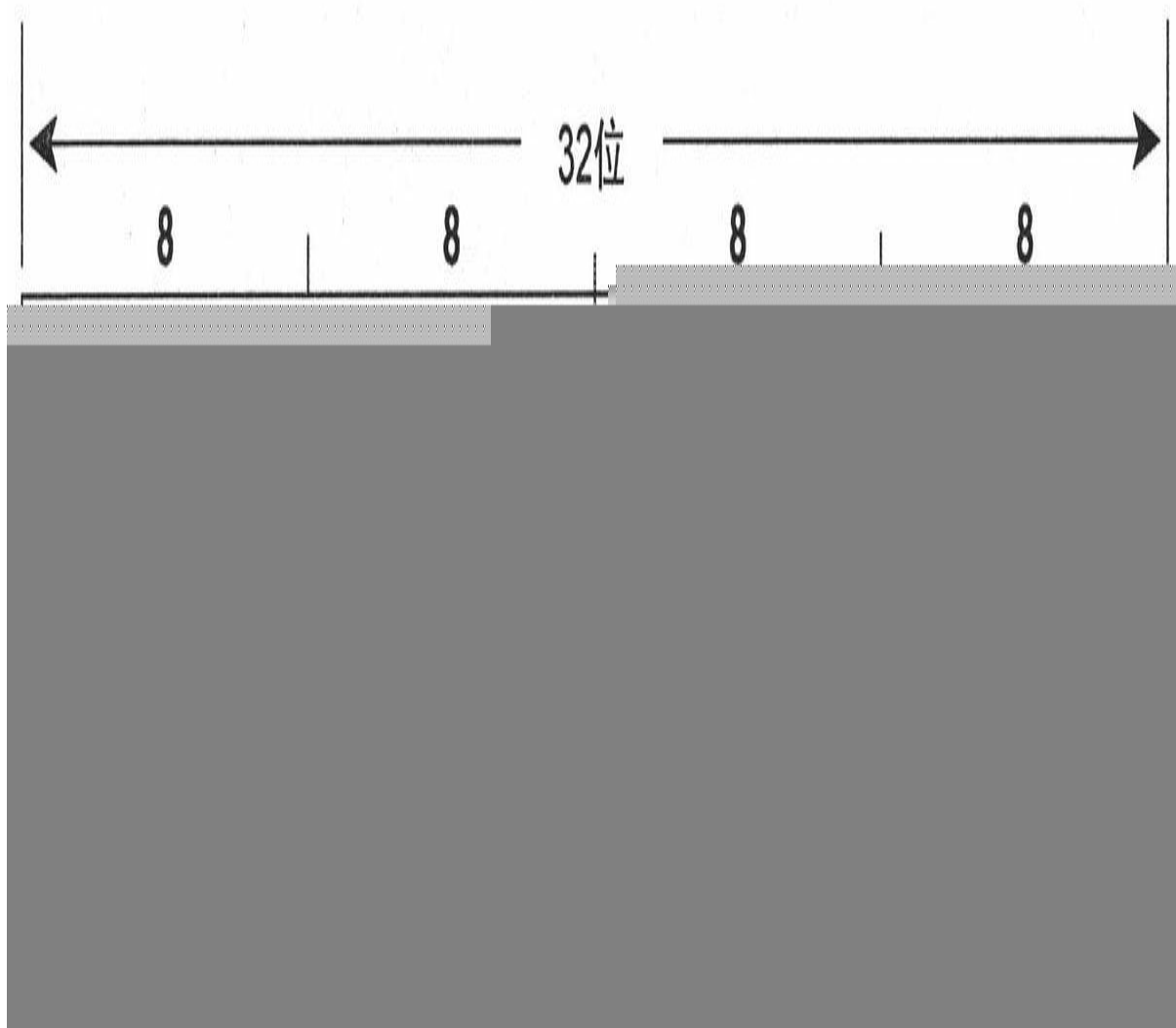


图4-9 6位循环地址空间

假设给定两个序列号48和18，由规则1可得48较新：

$48 > 18$ 且 $(48 - 18) = 30$ ，且 $30 < 32$ 。

假设给定两个序列号3和48，由规则2可得3较新：

$3 < 48$ 且 $(48 - 3) = 45$ ，且 $45 > 32$ 。

假设给定两个序列号3和18，由规则1可得18较新：

$18 > 3$ 且 $(18 - 3) = 15$ ，且 $15 < 32$ 。

可以看出规则强制序列号循环使用。

但是在运行状况不是很正常的网络上又会如何？设想在一个网络中使用6位序列号空间。现在其中的一台路由器决定离线，当它在离线之时，突然发送了3个相同且序列号为44（101100）的LSA。不幸的是，一个邻居也发生了故障——丢掉了几位数据，丢失的数据位是第2个LSA和第3个LSA序列号中的1位。随即该邻居路由器向外泛洪扩散了这3个LSA。结果造成3个LSA序列号各不相同。

44	(101100)
40	(101000)
8	(001000)

应用循环规则可得44比40更新，40比8更新，8又比44更新！这个结论将使3个LSA都持续扩散下去，数据库也不断地被最新的LSA更新，直到数据库缓冲被塞满，CPU超载为止，最终整个网络崩溃。

这一连锁事件听上去好像有些牵强，然而它确实是真实的。现代网络的前驱ARPANET早期曾经使用过带有6位序列号空间的链路状态协议。在1980年10月27号，两台路由器发生了上面所说的故障，从而造成ARPANET的停顿。[\[11\]](#)

（3）棒棒糖形序列号空间

这个古怪名称的结构是由Radia Perlman博士提出的。[\[12\]](#)棒棒糖形序列号空间是线性序列号空间和循环序列号空间的综合；如果你考虑一下，你会发现棒棒糖形序列号空间有一个线性组件和一个圆形组件。圆形空间的缺点是不存在一个数小于其他所有的数。线性空间的缺点是不能循环使用序列号，即序列号是有限的。

当路由器A重新启动时，它将从小于其他所有数的 a 开始。邻居将会识别出这个数，这时如果邻居数据库中还保留有上次序列号为 b 的LSA，那么它们会发送 b 给路由器A，则路由器A将跳至该序列号。在知道自己重启前使用的序列号之前，路由器A可能会发送不止一个LSA。因此，在邻居通知路由器A上次使用的序列号或上次使用的序列号从所有数据库中消失之前，必须准备充足的启动数以便路由器A不会用光所有序列号。

这些线性重启序列号形成了棒棒糖的棒。在这些数用完或邻居提供了使路由器A跳转的序列号之后，路由器A进入循环数空间，也就是棒棒糖的糖体部分。

在设计棒棒糖地址空间时使用了有符号数，即 $-k < 0 < k$ 。从 $-k$ 到1的负数形成棒棒，从0到 k 的正数形成循环空间。Perlman的序列号规则如下。假设给出两个数 a 、 b 及一个序列号空间 n 当且仅当：

- ① $a < 0$ 且 $a < b$ ，或
- ② $a > 0$ ， $a < b$ ，且 $(b-a) < n/2$ 或
- ③ $a > 0$ ， $b > 0$ ， $a > b$ ，且 $(a-b) > n/2$ 。

认为 b 比 a 更新。

图4-10给出了棒棒糖形序列号空间的一个实现。使用32位有符号数 n 可以产生 2^{31} 个正数和 2^{31} 个负数。 $-N$ （ -2^{31} 或0x80000000）和 $N-1$ （ $2^{31}-1$ 或0x7FFFFFFF）没有被使用。路由器在线时将从 $-N+1$ （0x80000001）开始使用序列号，一直增加到0，这时将进入循环数空间。当序列号到达 $N-2$ （0x7FFFFFFE）时，序列号将返回到0（不使用 $N-1$ ）。

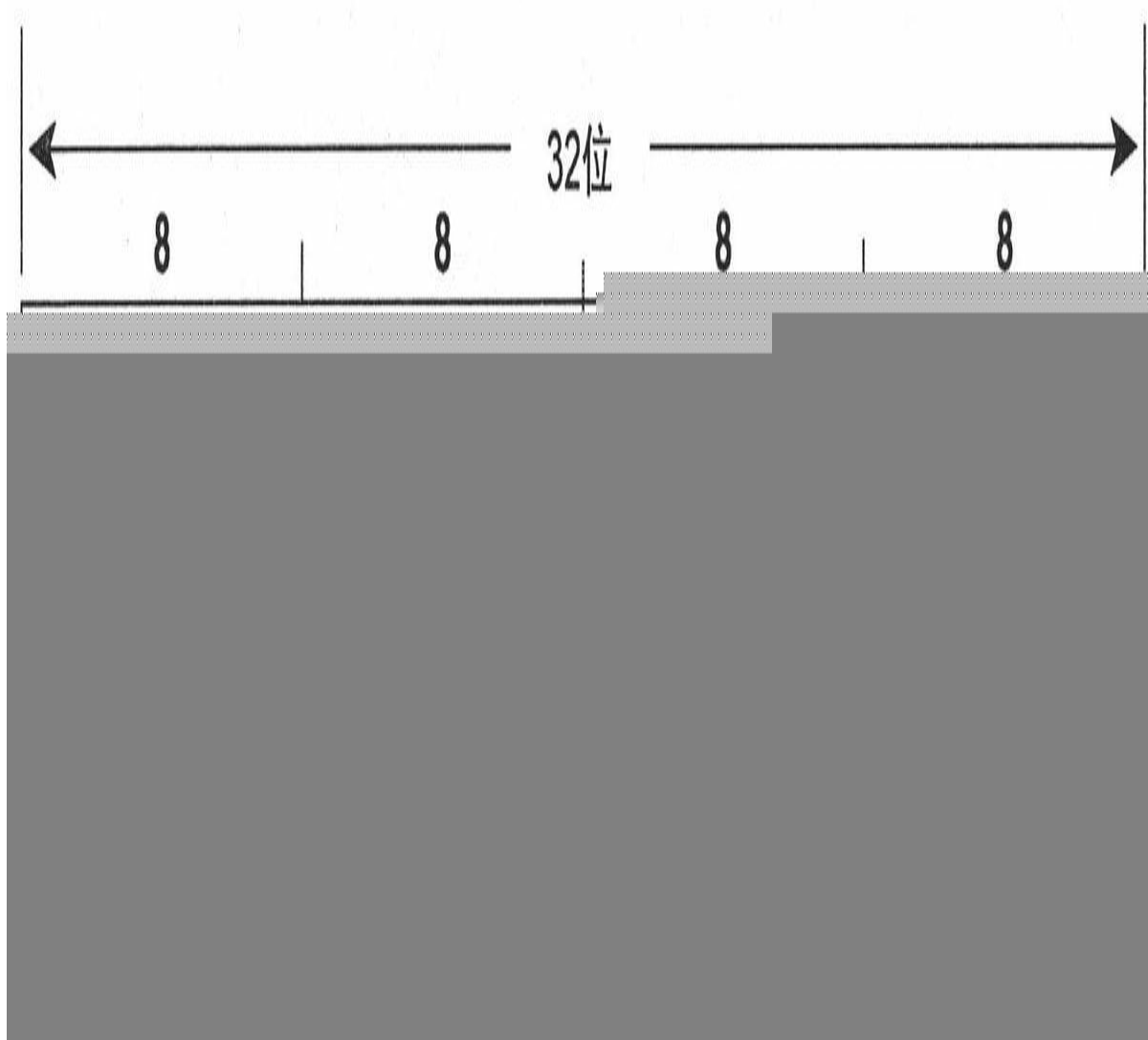


图4-10 棒棒糖形序列号空间

假设路由器再次重新启动，最后一个被发送的LSA序列号是0x00005de3（循环序列号空间的一部分）。路由器重启后，当它与邻居同步数据库时，路由器发送序列号为0x80000001（ $-N+1$ ）的LSA。邻居检查它的数据库发现该路由器重启前LSA的序列号为0x00005de3，这时邻居将把这个LSA发给重新启动的路由器，实际上是说，“这是你离开时的位置。”接着重启路由器记录下这个序列号为正数的LSA。如果下一时刻路由器需要发送新的LSA，它将使用序列号0x00005de6。

棒棒糖形序列号空间是用在OSPF的最初版本OSPFv1（RFC1131）中的。虽然使用有符号数是对线性数空间的一个改进，但是发现棒棒糖形

序列号空间的循环部分与纯粹的循环空间同样容易出现歧义。OSPFv1的部署一直处于试验阶段。当前使用的OSPF版本OSPFv2（最初见RFC 1247）采用了线性和棒棒糖形序列号空间的最好的特性。OSPFv2像棒棒糖形序列号数一样使用了从0x80000001开始的有符号数空间。但是当序列号数变为正数时，序列号空间持续保持线性直至到达最大数0x7FFFFFFF为止。此刻，在重启之前OSPF进程必须从所有链路状态数据库中删除LSA。

2. 老化（Aging）

LSA格式中将要包含一个用于通告年龄的字段。当LSA被创建时，路由器将该字段设置为0。随着数据包的扩散，每台路由器都会增加通告中的年龄。[\[13\]](#)

老化过程为泛洪扩散过程增加了另一层可靠性，该协议为网络定义了一个最大年龄差距（MaxAgeDiff）值。路由器可能接收到一个LSA的多个拷贝，其中序列号相同，年龄不同。如果年龄的差距小于MaxAgeDiff，那么认为是由于网络的正常时延造成了年龄的差异，因此数据库原有的LSA继续保存，新收到的LSA（年龄更大）不被扩散。如果年龄差距超过MaxAgeDiff，那么认为网络发生异常，因为新被发送的LSA的序列号值没有增加。在这种情况下，较新的LSA被记录下来，并将数据包扩散出去。典型的MaxAgeDiff值为15min（用于OSPF）。

若LSA驻留在数据库中，则LSA的年龄会不断增加。如果链路状态记录的年龄增加到某个最大年龄值（MaxAge）——由特定的路由选择协议定义——那么一个带有MaxAge值的LSA被泛洪扩散到所有邻居，邻居随即从数据库中删除相关记录。

当LSA的年龄到达MaxAge时，将被从所有的数据库中删除，这需要有一种机制来定期地确认LSA并且在达到最大年龄之前将它的计时器复位。链路状态刷新计时器（LSRefreshTime）就是做此用途的；[\[14\]](#)一旦计时器超时，路由器将向所有邻居泛洪扩散新的LSA，收到的邻居会把有关路由器记录的年龄设置为新接收到的年龄。OSPF定义MaxAge为1小时，LSRefreshTime为30min。

[4.3.3 链路状态数据库](#)

除了泛洪扩散LSA和发现邻居，链路状态路由选择协议的第3个主要任务是建立链路状态数据库。链路状态或拓扑数据库把LSA作为一连串记录保存下来。虽然LSA还包括序列号、年龄和其他信息，但这些变量主要用于管理泛洪扩散进程。对于最短路径的决策进程来说，通告路由器的ID、连接的网络和邻居路由器以及与网络 and 邻居相关联的代价是非常重要的信息。正如前面句子所隐含的，LSA还包括两类通用信息：[\[15\]](#)

- 路由器链路信息 ——使用三元组（路由器ID、邻居ID、代价）通告路由器的邻居路由器，这里的代价是指链路到邻居的代价。
- 末梢网络信息 ——使用三元组（路由器ID、网络ID、代价）通告路由器直接连接的末梢网络（没有邻居的网络）。

最短路径优先（SPF）算法对路由器链路信息进行一次计算以建立到每台路由器的最短路径，然后使用末梢网络信息向路由器添加网络。图4-11给出的网络包括路由器及路由器之间的链路，为简单起见没有给出末梢网络。注意，在几条链路两端的代价各不相同，因为代价与接口的出站方向有关。例如，从RB到RC的链路代价为1，但是对相同的链路，从RC到RB方向的代价却为5。

对于图4-11的网络，表4-2给出了一个通用的链路状态数据库，在每台路由器中都保存了一份拷贝。当你阅读这个数据库时，你将会发现数据库完整地描述了网络。现在通过运行SPF算法可以计算出一棵到达每台路由器的最短路径树。

表4-2 关于图4-11中网络的拓扑数据库

路由器ID	邻居	代价
RA	RB	2
RA	RD	4
RA	RE	4
RB	RA	2
RB	RC	1
RB	RE	10
RC	RB	5
RC	RF	2

RD
RD
RD
RE
RE
RE
RE
RE
RF
RF
RF
RG
RG
RH
RH

RA	4
RE	3
RG	5
RA	5
RB	2
RD	3
RF	2
RG	1
RH	8
RC	2
RE	2
RH	4
RD	5
RE	1
RE	8
RF	6

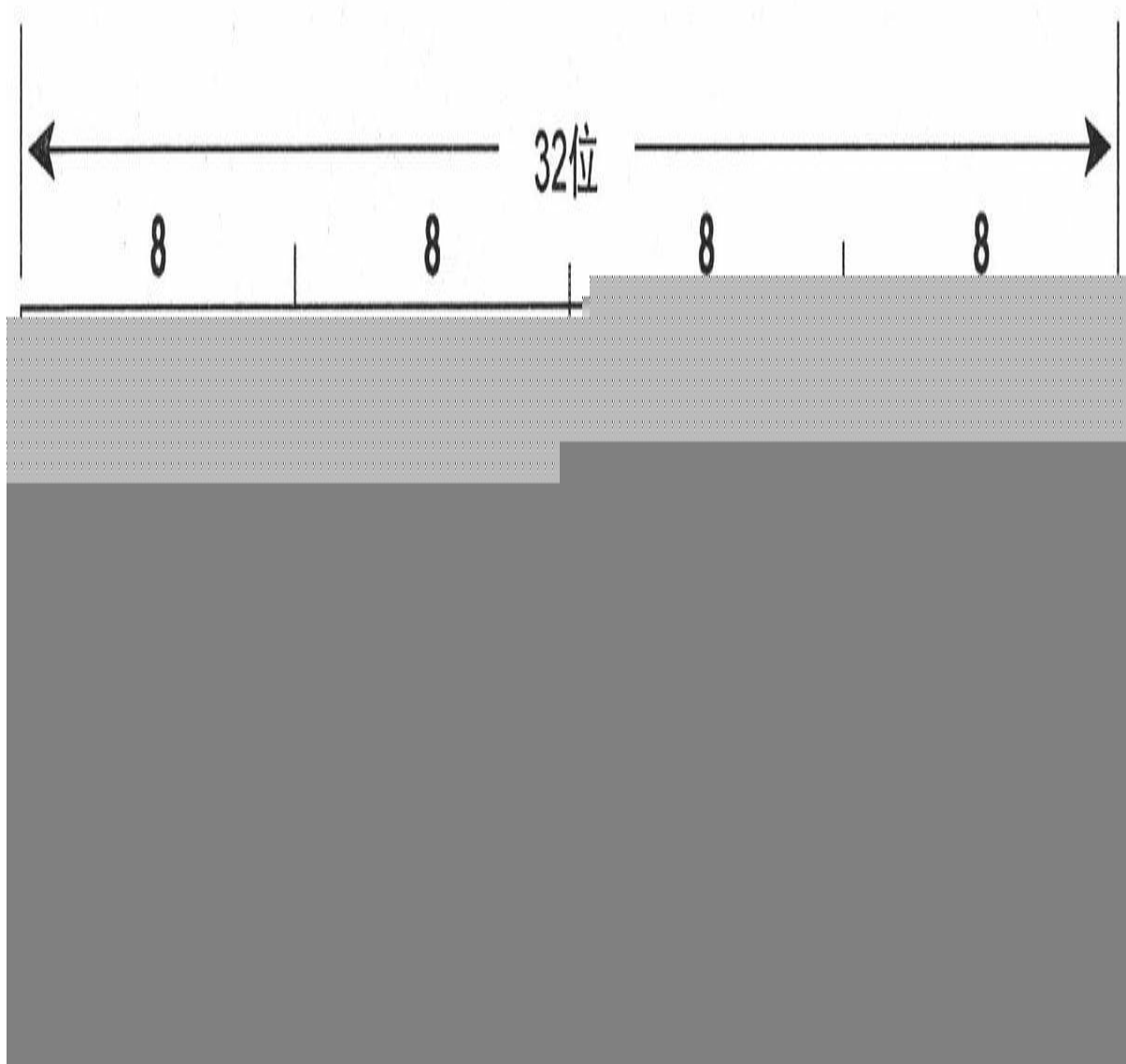


图4-11 链路代价是按照出站接口的方向计算的，并且在网络中所有接口的代价没有必要完全相同

4.3.4 SPF算法

在路由选择领域里，很不幸的是Dijkstra算法常常被认为是最短路径优先算法。毕竟每个路由选择协议的目标都是计算最短路径。另一个不幸的是Dijkstra算法常常被描述的比实际复杂得多，因为许多作者都使用集合论符号讨论它。最清晰的描述来自E.W.Dijkstra的原稿。这里将使用他的原话，并插入针对链路状态路由选择协议的解释：

构造一棵树[a]，使 n 个节点之间的总长最小（树是一个在每两个节点之间仅有一条路径的图）。

在我们给出的构造过程中，分枝被分成3个集合：

- I. 被明确分配给构造中的树的分枝（它们将在子树中）；
- II. 这个分枝的隔壁分枝被添加到集合I；
- III. 剩余的分枝（抛弃或不考虑）。

节点被分成2个集合：

- A. 被集合I中的分枝连接的节点；
- B. 剩余的节点（集合II中有且仅有一个分枝将指向这些节点中的每一个节点）。

下面我们开始构造树，首先选择任意一个节点作为集合A中仅有的成员，并将所有拿这个节点做端点的分枝放入集合II中。开始集合I是空的。然后我们重复执行下面两步。

步骤1：集合II中最短的分枝被移出并加入集合I。结果，一个节点被从集合B传送到集合A。

步骤2：考虑从这个节点（刚才被传送到集合A中的节点）通向集合B中节点的分枝。如果构建中的分枝长于集合II中相应的分枝，那么分枝被丢弃；否则，用它替代集合II中相应的分枝，并且丢弃后者。

接着我们回到第1步并重复此过程直到集合II和集合B为空。集合I中的分枝形成所要求的树。[\[16\]](#)

配合路由器的算法，第一点注意的是，Dijkstra描述了3个分枝集合：I、II和III。在路由器中，使用3个数据库表示3个集合：

- 树数据库 —— 树数据库用来表示集合I。通过向数据库中添加分枝实现向最短路径树中添加链路（分枝）。当算法完成时，这个数据库将可以描述最短路径树。

- 候选对象数据库 —— 候选对象数据库对应集合II。按照规定的顺序从链路状态数据库向该列表中复制链路，作为向树中添加的候选对象。
- 链路状态数据库 —— 按照前面的描述，这里保存所有链路，这个拓扑数据库对应集合III。

Dijkstra还指定了两个节点集合——A和B。这里的节点是路由器。这些路由器被明确地用路由器链路三元组（路由器ID、邻居ID、代价）中的邻居ID表示。集合A由树数据库中链路所连接的路由器组成。集合B是所有其他的路由器。由于全部要点是发现到每台路由器的最短路径，所以当算法结束时集合B应该为空。

下面是一台路由器中采用的Dijkstra算法版本：

步骤1： 路由器初始化树数据库，将自己作为树的根。这表明路由器作为它自己的邻居，代价为0。

步骤2： 在链路状态数据库中，所有描述通向根路由器邻居链路的三元组被添加到候选对象数据库中。

步骤3： 计算从根到每条链路的代价，候选对象数据库中代价最小的链路被移到树数据库中。如果两个或更多的链路离根的最短代价相同，选择其中一条。

步骤4： 检查添加到树数据库中的邻居ID。除了邻居ID已在树数据库中的三元组之外，链路状态数据库中描述路由器邻居的三元组被添加到候选对象数据库中。

步骤5： 如果候选对象数据库中还有剩余的表项，回到第3步。如果候选数据库为空，那么终止算法。在算法终止时，在树数据库中，每个单一的邻居ID表项将表示1台路由器，到此最短路径树构造完毕。

表4-3总结了应用Dijkstra算法为图4-11中的网络构建最短路径树的过程和结果。路由器RA正在运行算法，并使用表4-2中的链路状态数据库。图4-12给出了通过该算法为路由器RA构造的最短路径树。在每台路由器完成自己最短路径树的计算之后，它会检查其他路由器的网络链路信

息，并且相当容易地把末梢网络添加到树中。根据这些信息，表项可以被制作成路由表。

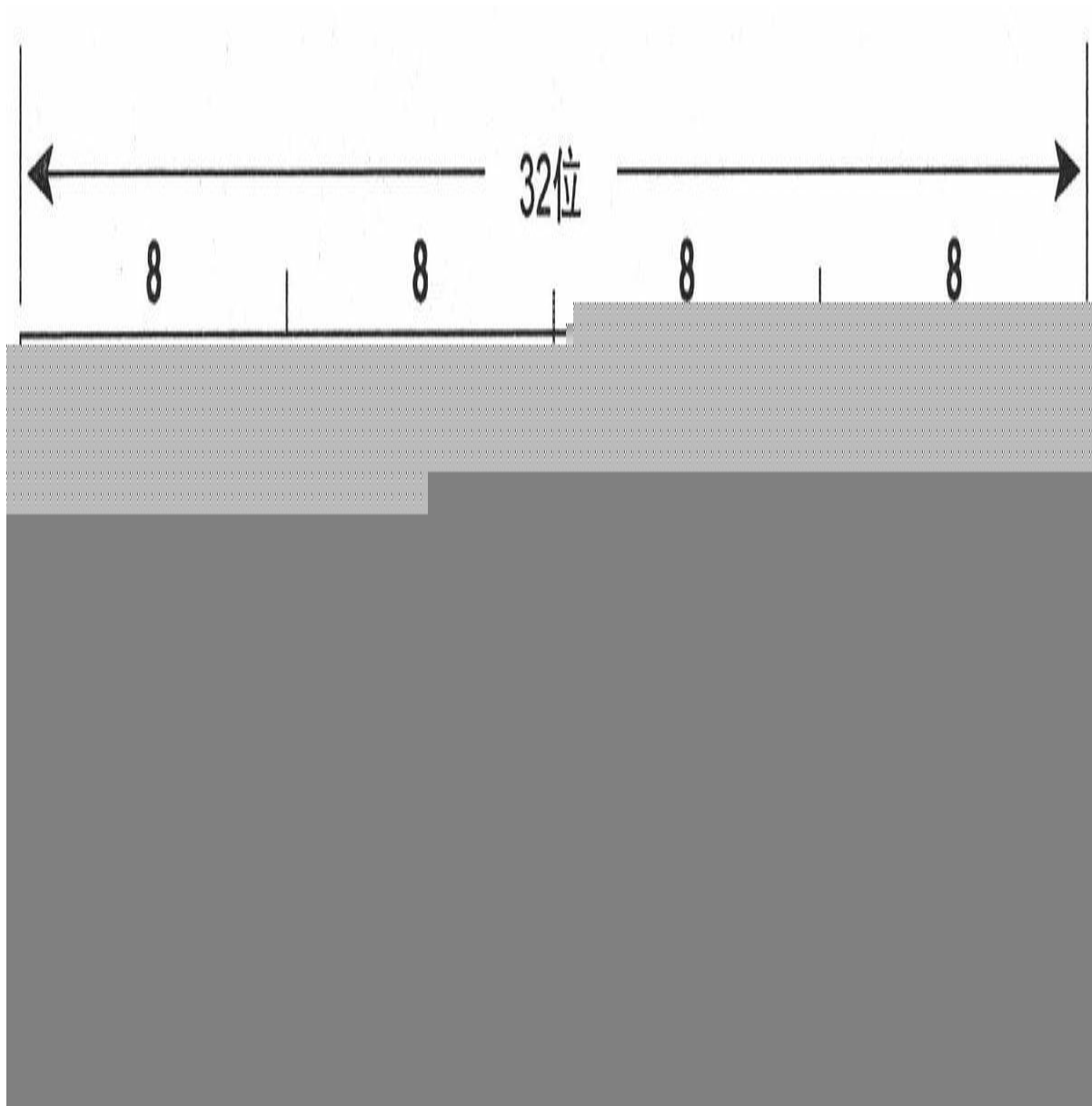


图4-12 用表4-3中的算法得到的最短路径树

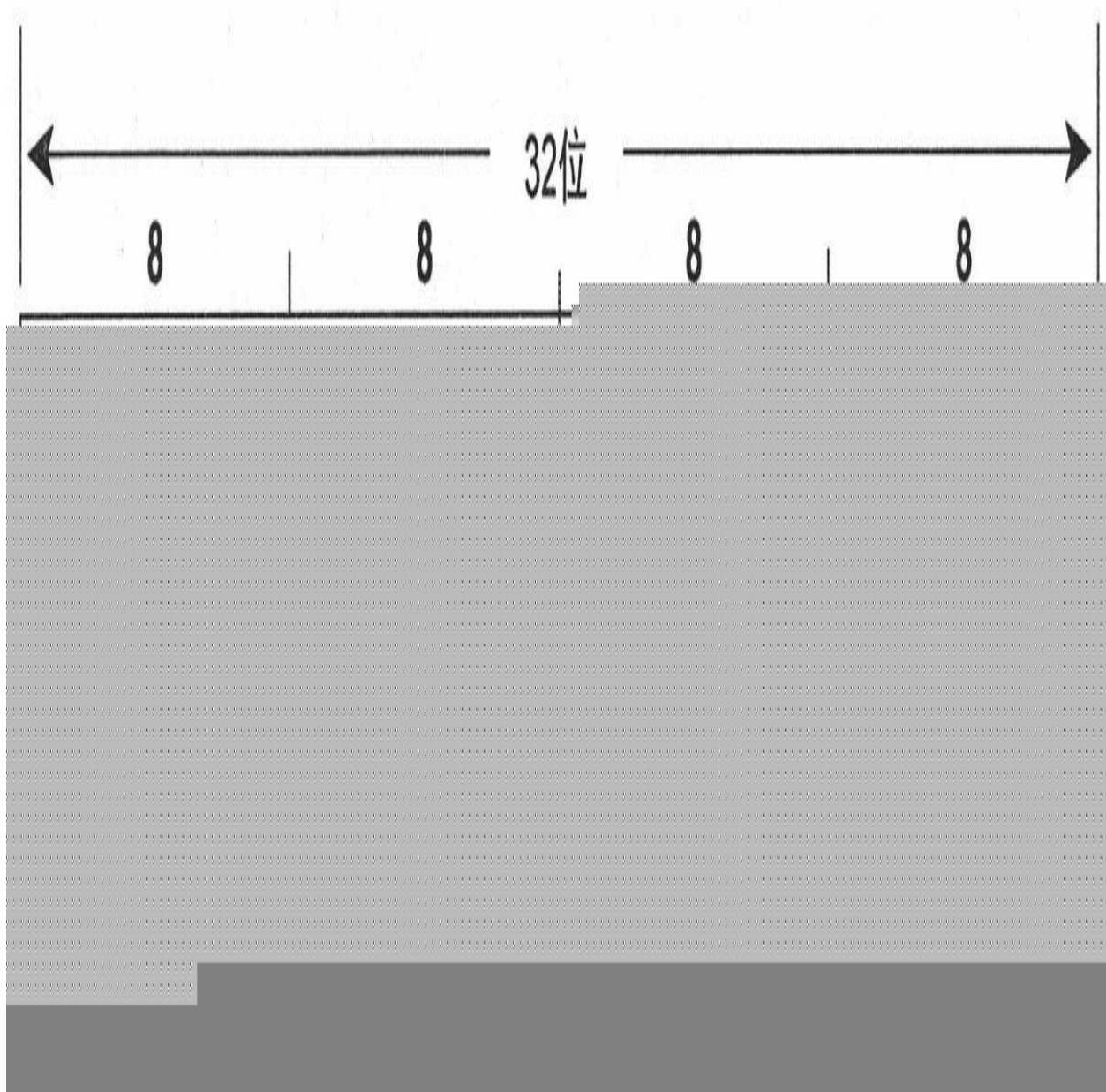
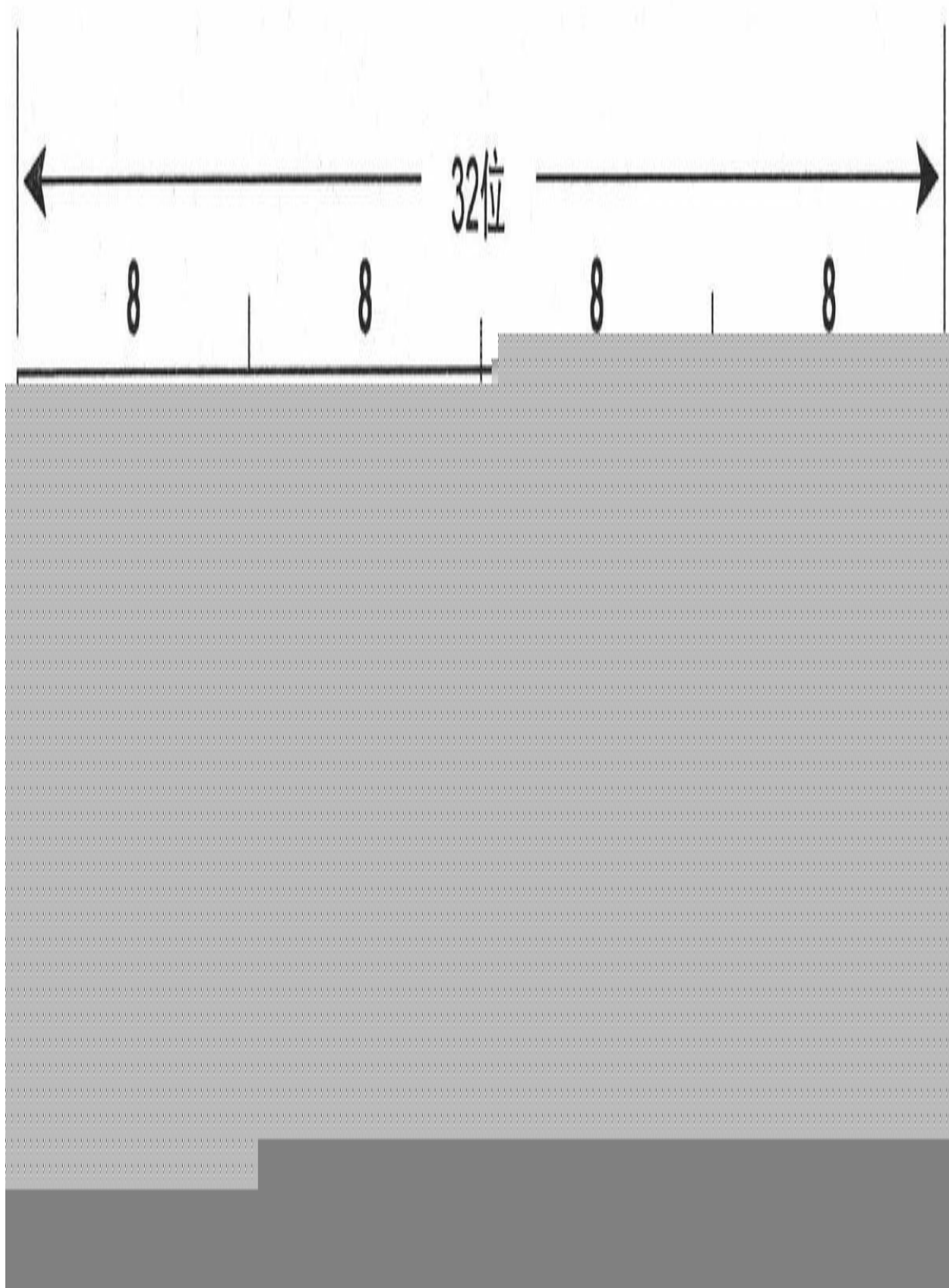


表4-3 对表4-2的数据库应用Dijkstra算法



4.3.5 区域

一个区域是构成一个网络的路由器的一个子集。将网络划分为区域是针对链路状态协议的3个不利影响所采取的措施。

- 必要的数据库要求内存的数量比距离矢量协议更多。
- 复杂的算法要求CPU时间比距离矢量协议更多。
- 链路状态泛洪扩散数据包对可用带宽带来了不利的影响，特别是不稳定的网络。

目前设计的链路状态协议和使用该协议的路由器都能够减少这些影响，但是却没有消除这些影响。在最后一节我们在含有8台路由器的网络中，分析了链路状态数据库看上去是什么样，SPF算法是如何工作的。但要记住，对于连接到8台路由器上的末梢网络，并由此形成的SPF树的叶节点，在这里没有考虑。现在我们假设一个有8000台路由器的网络，你就能理解对内存、CPU和带宽所造成影响的利害关系了。

正如图4-13所示，通过划分区域可以减小这些影响。当一个网络被划分为多个区域时，在一个区域内的路由器仅需要在本区域扩散LSA，因而只需要维护本区域的链路状态数据库。数据库越小，意味着需要内存越少，运行SPF算法需要的CPU周期也越少。如果拓扑改变频繁发生，引起的扩散将被限制在不稳定的区域。

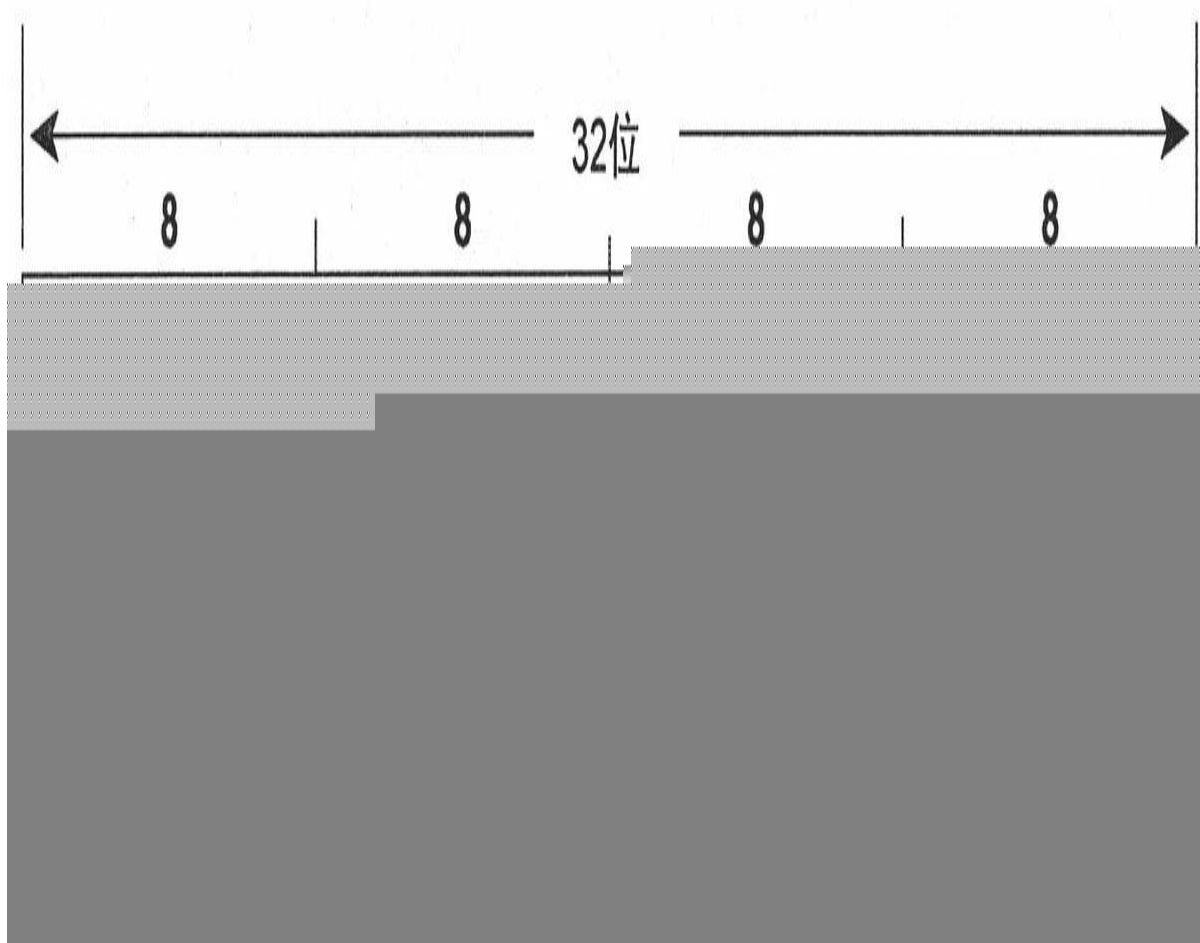


图4-13 区域的使用减少了链路状态对系统资源的需求

区域边界路由器是连接两个区域的路由器，它属于所连接的两个区域，而且必须为每个区域维护各自的拓扑数据库。正像在网络上发送数据包到其他网络的主机仅需要知道如何找到本地路由器一样，在一个区域内想往其他区域发送数据包的路由器仅需要知道怎样找到本地区域边界路由器。换句话说，区域间路由器/区域内路由器之间的关系就如同主机/路由器之间的关系，除了所在层次更高一些。

距离矢量协议，如RIP和IGRP，不使用区域。假设这些协议除了把一个大型的网络看成一个单一的实体外，没有任何依靠，那么它们就必须计算出到每个网络的路由，并且每30s或90s就必须广播这个巨大的路由表。很明显，利用区域的链路状态协议可以节省系统资源。

4.4 内部和外部网关协议

区域向网络体系结构中引入了一个层次，在此基础上，将一组区域组成一个更大的区域又向网络体系结构中引入了另一个层次。这些更高层的区域在IP领域内叫自主系统，在ISO模型中叫路由选择域。

一个自主系统被定义为在共同管理域下的一组运行相同路由选择协议的路由器。假设现代网络的活动存在变移性，那么该定义的后半部分将不是非常准确。部门、分公司乃至整个公司常常会合并，随着它们的合并原来设计使用不同路由选择协议的网络也合并在一起。结果是，许多网络使用多种不精确的等级将多种路由选择协议结合在一起，并处于共同的管理之下。所以自主系统的当前定义应该是在共同管理下的网络。

在一个自主系统内运行的路由选择协议称为内部网关协议（IGP）。在本章的例子中出现的所有距离矢量和链路状态协议都是IGP。

在自主系统之间或路由选择域之间的路由选择协议称为外部网关协议（EGP）。IGP发现网络之间的路径，而EGP发现自主系统之间的路径。下面这些都是EGP：

- 边界网关协议（BGP）。
- IP外部网关协议（EGP）（是，一个名为EGP的EGP）。
- ISO的域间路由选择协议（IDRP）。

Novell也把EGP功能合并到NLSP中，叫第3层路由选择。

在给出这些定义的同时，还必须要提到的是，术语自主系统（autonomous system）的通常用法并不是绝对的。各种标准文档、文献以及人们都趋向于给这个术语多种含义；其结果是，对于理解听到或读到该术语的上下文环境是十分重要的。

本书在两种上下文环境中使用自主系统：

- 如本节开始定义的，自主系统可以指路由选择域。在该上下文环境中，一个自主系统是一个有一个或多个IGP的系统，它完全自

自主于其他IGP系统。在这些自主系统之间使用EGP。

- 自主系统也可以认为是一个过程域，或一个IGP进程，它自主于其他IGP进程。例如，使用OSPF的路由器可以被认为是OSPF自主系统。EIGRP的也可以在这种上下文环境中使用自主系统。在这些自主系统之间使用了路由的重新分配。

在本书中，上下文环境将指明在不同地方所讨论的自主系统是属于哪一种形式的自主系统。

4.5 静态或动态路由选择

在阅读完动态路由选择协议的显赫细节之后，留下的印象一定是动态路由选择协议比静态路由选择协议好。动态路由选择协议的主要任务是自动监测和适应网络拓扑的变化，记住这一点很重要。但是，为实现“自动化”需要在带宽、队列空间、内存和处理时间上付出代价。

使用动态路由选择协议的另一个代价是减少了对路由选择的控制。对于一个指定的目标网络，路由选择协议可以代替我们确定最佳路径。如果需要进行精确的路由选择，特别是当你期望的路径和路由选择协议选择的不一致时，还是使用静态路由选择更好一些。

静态路由选择的最大缺陷是管理困难。这对于存在许多可选路由的中型和大型网络来说是真实的，但对于几乎没有可选路由的小型网络来说就不适用了。

如图4-14所示，在较小的网络中很流行星型（hub-and-spoke）拓扑。如果到路由器的一个分支发生中断，是否有另一条路由供动态路由选择协议选择？因此对于这种网络来说，十分适合使用静态路由选择协议。在中心路由器上为每个分支上的路由器配置一条静态路由，而在每台分支路由器上配置一条指向中心路由器的缺省路由，这样网络就可以工作了（缺省路由将在第12章中介绍）。

在设计网络时，最简单的解决方法常常是最好的办法。在确定静态路由选择协议不能满足设计要求之后，可以选择动态路由选择协议。

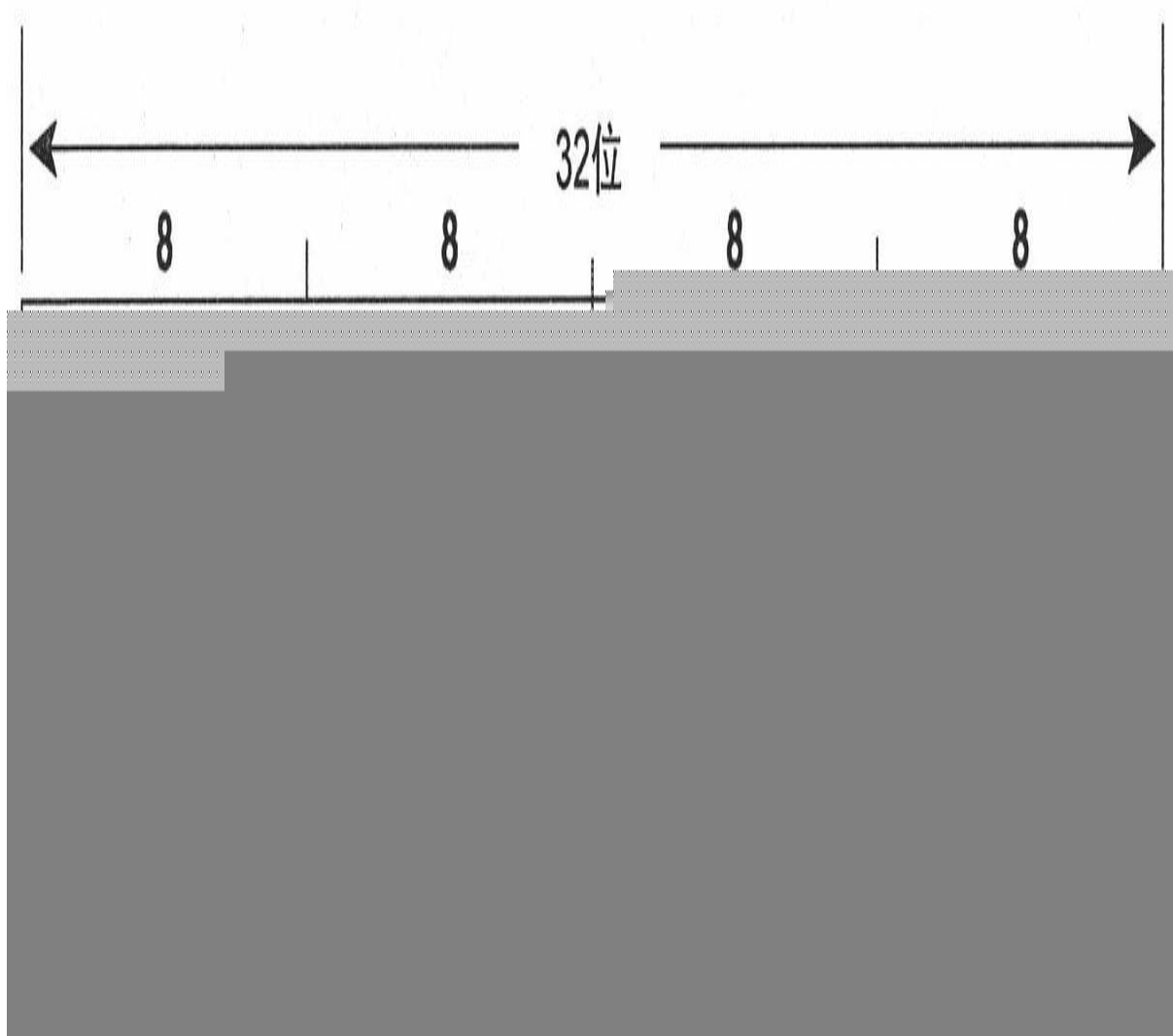


图4-14 星型网络适合使用静态路由选择

4.6 展 望

既然动态路由选择协议的基础知识已经分析完了，现在该是讨论具体的路由选择协议的时候了。下一章是**RIP**，一种最古老最简单的动态路由选择协议。

4.7 推荐读物

Perlman, R. *Interconnections: Bridges and Routers*. Reading, Massachusetts: Addison-Wesley;1992.

4.8 复习题

1. 什么是路由选择协议？
2. 路由选择算法执行的基本过程是什么？
3. 为什么路由选择协议使用度量？
4. 什么是收敛时间？
5. 什么是负载均衡？给出负载均衡的4种不同类型。
6. 距离矢量路由选择协议是什么？
7. 指出距离矢量协议存在的几个问题。
8. 邻居是什么？
9. 路由失效计时器的作用是什么？
10. 解释简单水平分隔与毒性逆转水平分隔的区别？
11. 什么是计数到无穷大问题？如何对其进行控制？
12. 什么是抑制计时器？它们如何工作？
13. 距离矢量和链路状态路由选择协议的区别是什么？
14. 拓扑数据库的作用是什么？
15. 解释链路状态网络中收敛所包含的基本步骤。
16. 在链路状态协议中序列号为什么重要？
17. 在链路状态协议中老化服务的作用是什么？
18. 解释SPF算法是怎样工作的。

19. 区域对链路状态网络的好处是什么？

20. 什么是自主系统？

21. IGP和EGP的区别是什么？

[1] 在这8种协议中，BGP已经取代了EGP，Cisco Systems公司的EIGRP也取代了它自己的IGRP，RIPv2也正在迅速取代RIPv1。

[2] 许多点到点链路都被配置为“未编号”的链路——也就是不给点到点链路两端的接口分配地址。但是这又不违背每个接口都必须有地址这个原则，因为这些接口通常使用路由器上的环路地址作为代理地址。

[3] R.E.bellman。 *Dynamic Programming*。 普林斯顿，新泽西：普林斯顿大学出版；1957。

[4] L.R.Ford Jr.和D.R.Fulerson。 *Flows in Networks*。 普林斯顿，新泽西：普林斯顿大学出版；1962。

[5] Cisco的增强型IGRP明显不同于这种协定。EIGRP虽然是距离矢量协议，但是它既不定期发送更新信息，也不进行广播，而且更新信息也不是整个路由表。第7章将会讨论EIGRP。

[6] 这个表述不完全正确。在某些实现中主机也监听路由更新信息；但是在这个讨论中重要的是路由器如何工作。

[7] 一般是拿路标做比喻。你可以在Radia Perlman的 *Interconnections* 一书中找到一个好的表达。

[8] S.Floyd和V.Jacobson。“周期性路由选择消息的同步。”ACM Sigcomm'93 Symposium, 1993年9月。

[9] 就是所有的路由器使用相同的路由选择协议。

[10] LSA是一个OSPF术语，相应由IS-IS建立的数据单元叫链路状态PDU（LSP）。但对于常规讨论，LSA更适合我们的需要。

[11] E.C.Rose。“网络控制协议的脆弱性：一个例子。”计算机通信回顾，1981年7月。

[12] R.Perlman。“路由选择信息的容错广播。”计算机网络，第7卷，1983年12月，395~405页。

[13] 当然，另一个选项是从某个最大年龄开始，然后递减。OSPF是递增；IS-IS是递减。

[14] LSRefreshTime、MaxAge和MaxAgeDiff是OSPF架构中的常量。

[15] 实际上，信息的种类不止两种，而且还包括多种链路状态数据包类型。这些在具体路由选择协议的章节中会提到。

[16] E.W.Dijkstra。“关于连通图中两个问题的注释。”Numerische Mathematik。第一卷，1959年，269~271页。

第二部分

内部路由选择协议

第5章 路由选择信息协议（RIP）

第6章 RIPv2、RIPng和无类别路由选择

第7章 增强型内部网关路由选择协议（EIGRP）

第8章 开放最短路径优先协议（OSPFv2）

第9章 OSPFv3

第10章 集成IS-IS协议

本章包括以下主题：

- RIP的基本原理与实现；
- 配置RIP；
- RIP故障诊断。

第5章

路由选择信息协议

(RIP)

RIP协议作为最早的距离矢量型IP路由选择协议依然被广泛地使用着，当前存在着两个版本。本章介绍的是RIP协议的第1版，第6章中包括了RIP协议第2版的内容，后者对RIPv1的功能作了部分增强。版本1和版本2最主要的区别是，RIPv1是有类别路由选择协议，而RIPv2是无类别路由选择协议。本章介绍有类别路由选择，第6章介绍无类别路由选择。同时，第6章也介绍了RIPng协议，这是RIPv2协议为了支持IPv6协议而设计的修订版本。

距离矢量协议基于Bellman^[1]、Ford和Fulkerson^[2]开发的路由选择算法，一些早期网络在1969年就已经实现了这个协议，例如ARPANET和CYCLADES等网络。在20世纪70年代中期，Xerox公司开发了一种称为，PARC^[3] Universal Protocol的协议也简称为PUP协议，运行在它的可以说是现代以太网前身的3Mbit/s带宽的试验网络上。PUP使用网关信息协议（GWINFO）来路由，并发展成为Xerox网络系统（XNS）协议簇。同时，GWINFO发展成为XNS路由选择信息协议（XNS RIP）。XNS RIP也随之成为一些常用的路由选择协议的前身，如Novell的IPX RIP、AppleTalk的RTMP（路由选择表维护协议），以及IP RIP。

1982年，伯克利发布的UNIX 4.2BSD版中，通过一个称为“routed”的守护进程实现了RIP协议；很多后来的UNIX版本都是基于流行的UNIX 4.2BSD版本的，并且也都通过一个称为“routed”或“gated”^[4]的进程支持RIP协议。有意思的是，直到1988年才发布了一个RIP协议的标准，那时RIP已经被广泛应用。这个标准就是RFC 1058，由Charles Hedrick撰写，是RIP协议第1版的比较正式的标准。

由于读者阅读的文献不同，从而对RIP协议褒贬不一。RIP协议虽然没有后来一些路由选择协议功能强大，但它简单易用，已被广泛的应用，这意味着RIP协议在实际网络的实施中碰到的兼容性问题会比较少。在小型网络数据链路中，RIP协议设计的相当单一。在这些限定的条件下，

尤其是许多UNIX环境下，RIP协议依然是一个受欢迎的路由选择协议。

5.1 RIP的基本原理与实现

RIP协议的处理是通过UDP 520端口来操作的。所有的RIP消息都被封装在UDP用户数据报协议中，源和目的端口字段的值被设置为520。RIP定义了两种消息类型：请求消息（request messages）和响应消息（response messages）。请求消息用来向邻居路由器发送一个更新（update），响应消息用来传送路由更新。RIP的度量是基于“跳”数（hop count）的，1跳表示是与发出通告的路由器相直连的网络，16跳表示网络不可到达。

开始时，RIP从每个启用RIP协议的接口广播出带有请求消息的数据包。接着，RIP程序进入一个循环状态，不断地侦听来自其他路由器的RIP请求或响应消息，而接收请求的邻居路由器则回送包含它们的路由表的响应消息。

当发出请求的路由器收到响应消息时，它将开始处理附加在响应消息中的路由更新信息。如果路由更新中的路由条目是新的，路由器则将新的路由连同通告路由器的地址一起加入到自己的路由表中，这里通告路由器的地址可以从更新数据包的源地址字段读取。如果网络的RIP路由已经在路由表中，那么只有在新的路由拥有更小的跳数时才能替换原来存在的路由条目。如果路由更新通告的跳数大于路由表已记录的跳数，并且更新来自于已记录条目的下一跳路由器，那么该路由将在一个指定的抑制时间段（holddown period）内被标记为不可到达。如果在抑制时间超时后，同一台邻居路由器仍然通告这个有较大跳数的路由，路由器则接受该路由新的度量值。^[5]

5.1.1 RIP的计时器和稳定性

路由器启动后，平均每隔30s从每个启动RIP协议的接口不断地发送响应消息。除了被水平分隔法则抑制的路由条目之外，响应消息（或称为更新消息）包含了路由器的整个路由表。这个周期性的更新由更新计时器（update timer）进行初始化，并且包含一个随机变量用来防止表的同步。^[6]结果，一个典型的RIP处理单个更新的时间大约是25~35s。

RIP_JITTER是Cisco IOS中专有的一个随机变量，它缩短到一般更新时间的15%（即4.5s）。因此，Cisco路由器的更新时间在25.5~30s之间变化（如图5-1所示）。路由更新的目的地是到所有主机的广播地址

255.255.255.255。 [\[7\]](#)

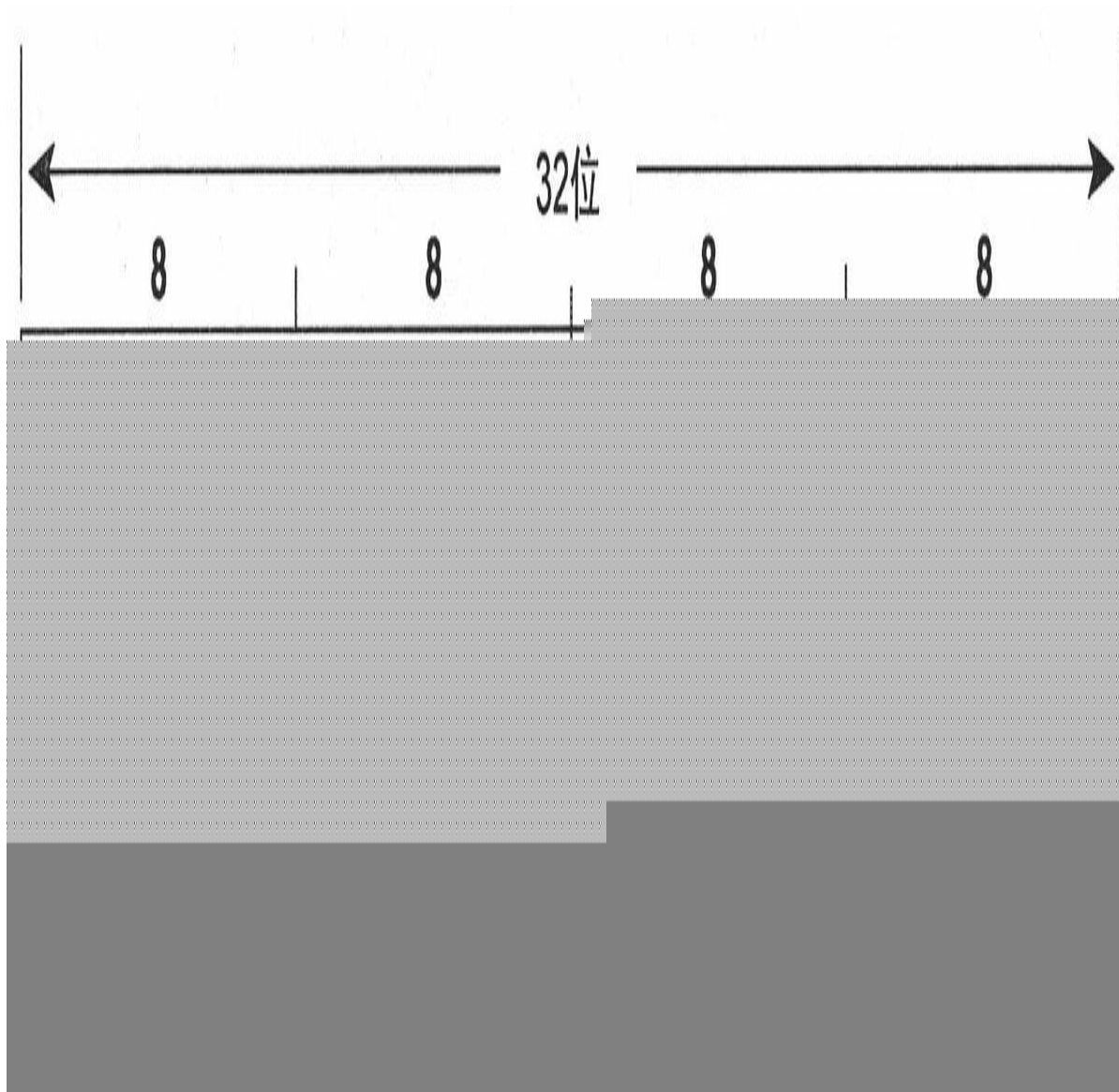


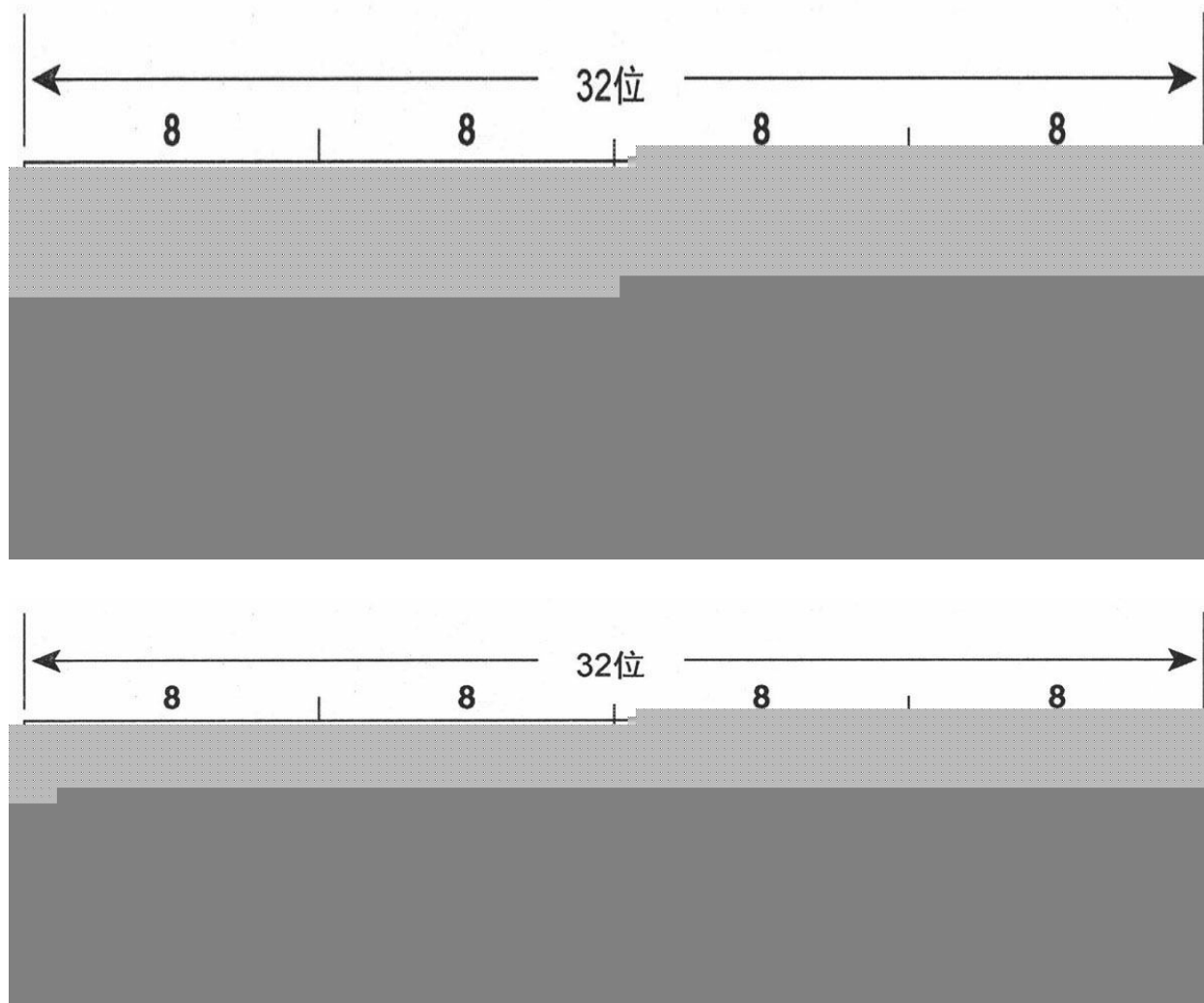
图5-1 RIP协议在每次重置计时器时，会增加一个小随机变量到更新计时器以避免路由表的同步。在Cisco路由器中，RIP的更新时间在25.5~30s之间变化，从图中可以看出这些更新的增量时间

RIP也使用一些其他的计时器。回忆一下第4章，距离矢量协议用到的无效计时器（invalidation timer）用来限制停留在路由表中的路由未被更新的时间。RIP称这个计时器为限时计时器（expiration timer）或超时计时器（timeout timer）；在Cisco IOS中，称为无效计时器（invalid timer）。无论什么时候，当有一条新的路由被建立，超时计时器就会被

初始化为180s，而每当接收到这条路由的更新消息时，超时计时器又将被重置成计时器的初始化值，即180s。如果一条路由的更新在180s（6个更新周期）内还没有收到，那么这条路由的跳数将变成16，也就是标记为不可到达的路由。

另一种计时器，称为垃圾收集（garbage collection）或刷新计时器（flush timer），它们所设置的时间长度一般比限时计时器的时间长240~60s。[\[8\]](#)如果垃圾收集计时器也超时了，则该路由将被通告为一条度量值为不可到达的路由，同时从路由表中删除该路由项。示例5-1显示了路由表中有一条被标记为不可达的路由，但还没有被刷新清除。

示例5-1 这台路由器经过6个更新周期的时间还没有收到关于子网10.3.0.0的更新。从而将这条路由标记为不可到达，但还没有被从路由表中刷新清除



第3个计时器是抑制计时器（holddown timer）。虽然RFC 1058没有关于Holddowns的介绍，但在Cisco路由器中运行的RIP协议使用了它们。如果一条路由更新的跳数大于路由表已记录的该路由的跳数，那么将会引起该路由进入长达180s（即6个路由更新周期）的抑制状态阶段。

这4个计时器可以通过下面的命令来操作：

timers basic update invalid holddown flush

该命令适用于RIP协议整个进程的运行处理。如果一台路由器的计时被改变了，那么这个RIP域中的所有路由器的计时都必须改变。因此，如果没有特别的原因，不应该改变这些计时器的缺省值。

RIP使用带毒性逆转（poison reverse）的水平分隔（split horizon）和触发更新（triggered updates）。不像普通的定期更新，触发更新只要有路由的度量值发生改变时就会产生，而且触发更新不会引起接收路由器重置它们的更新计时器；因为如果这么做的话，网络拓扑的改变会导致很多路由器在同一时间重置，从而引起定期的路由更新变得同步。为了避免拓扑改变后造成触发更新“风暴”，还需要使用另外一个计时器。当一个触发更新传播时，这个计时器被随机的设置为1~5s之间的数值；在这个计时器计时超时前不能发送并发的触发更新。

一些主机可以在“静”模式下使用RIP。这些所谓的“静”主机不产生RIP的更新消息，而只侦听RIP的更新消息，从而更新它们自己的路由表。举一个例子，在一台UNIX主机上可以使用带“-q”选项的“routed”启动“静”模式下的RIP。

5.1.2 RIP消息格式（RIP Message Format）

RIP的消息格式如图5-2所示。每条消息包含一条命令（Command）、一个版本号和路由条目（最大25条）。每个路由条目包括地址族标识（address family identifier）、路由可达的IP地址和路由的跳数。如果某台路由器必须发送大于25条路由的更新消息，那么必须产生多条RIP消息。注意，RIP消息的开始部分（头部）占用4个八位组字节（octets），而每个路由条目占用20个八位组字节。因此，RIP消息的大

小最大为 $4 + (25 \times 20) = 504$ 个八位组字节，再加上8个字节的UDP头部，RIP数据报的大小（不含IP包的头部）最大可达512个八位组字节。

- 命令（**Command**）——取值1或2，1表示该消息是请求消息，2表示该消息是响应消息。其他的取值都不被使用或保留用作私有用途。
- 版本号（**Version**）——对于RIPv1，该字段的值设置为1。
- 地址族标识（**Address Family Identifier, AFI**）——对于IP该项设置为2。只有一个例外情况，该消息是路由器（或主机）整个路由表的请求，这将在后面的章节讨论。
- IP地址（**IP Address**）——路由的目的地址。这一项可以是主网络地址、子网地址或主机路由地址。在5.1.4小节，将说明如何区分这3种类型的路由。
- 度量（**Metric**）——正如前面章节所说，在RIP中指跳数，该字段的取值范围在1~16之间。

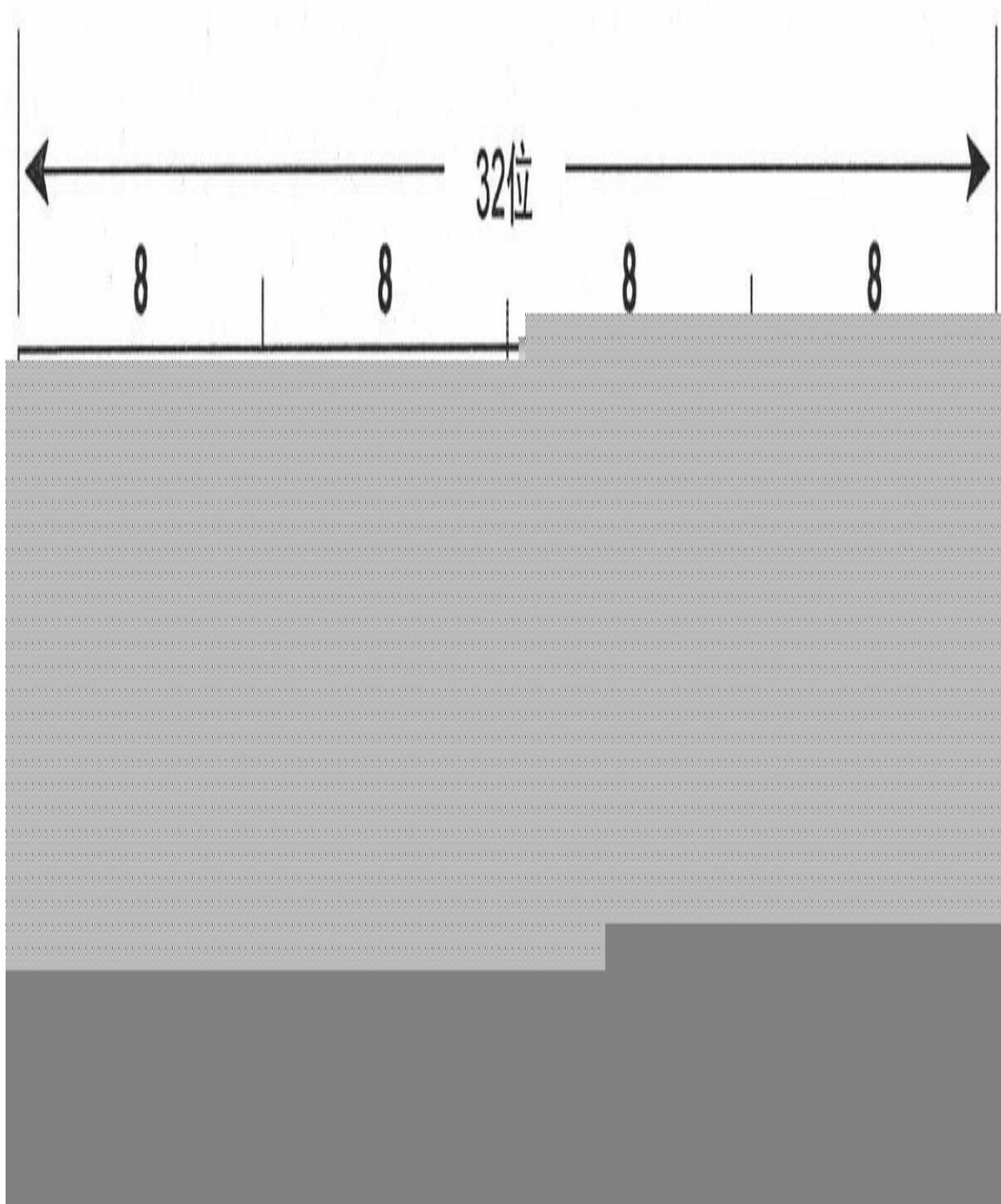


图5-2 RIP的消息格式

图5-3显示了用协议分析仪观察到的一个RIP消息的解码。

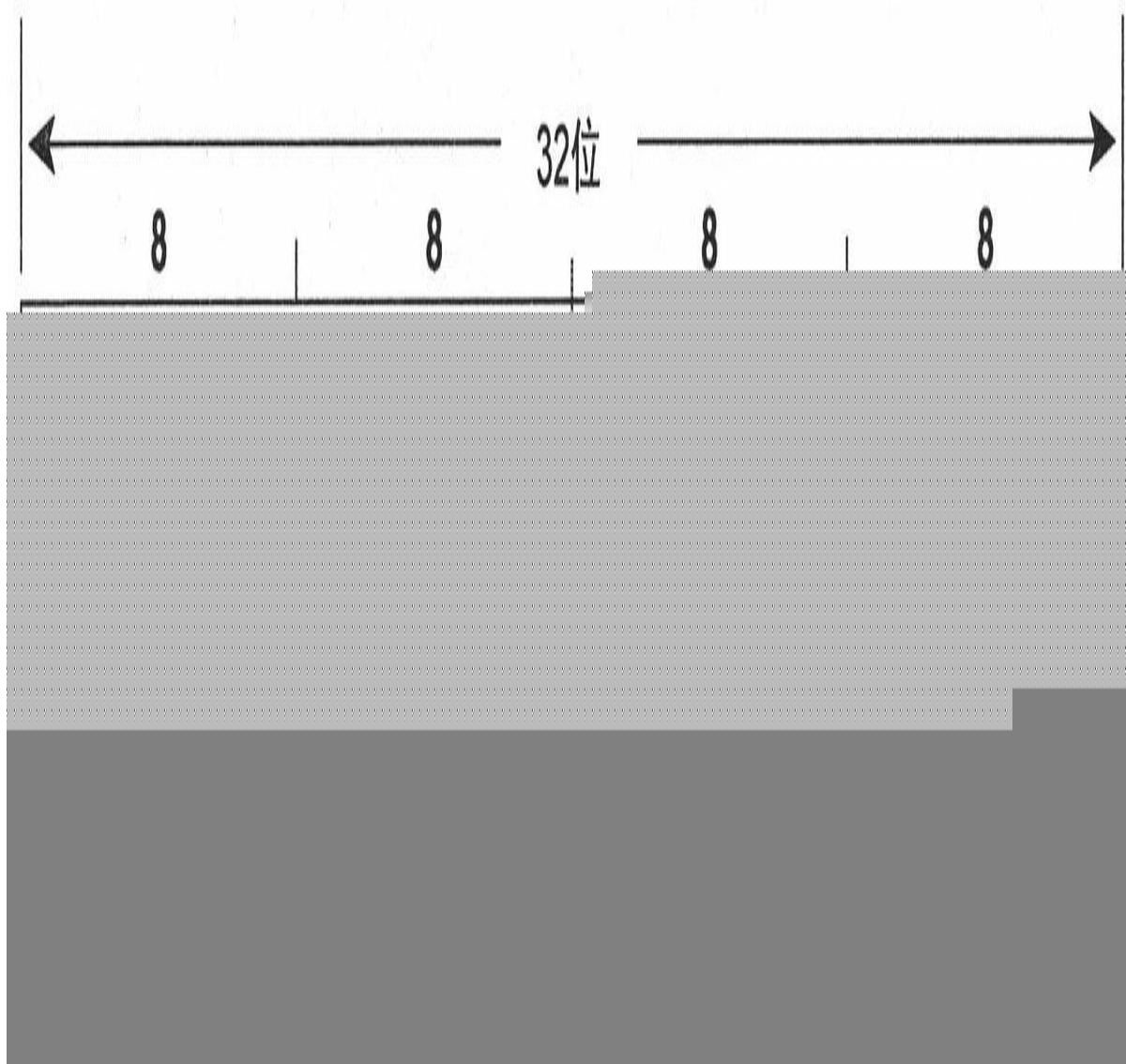


图5-3 从协议分析仪可以看出，RIPv1不使用子网掩码和下一跳字段。这些字段在RIPv2中使用，将在第7章中讲述

由于一些历史的影响造成RIP消息的格式不尽合理，消息格式中没有使用的比特空间远远大于所使用的。这些影响从RIP最初发展到后来就都存在，RIP最初来自于XNS协议，开发人员试图使它适合更广泛的地址族；在BSD的影响下，它所使用的套接字（socket）地址需要RIP的字段增大到32位字的边界长度。无论初衰如何，读者在第6章中将会看到这些未用的字段后来具有很大的用处。

5.1.3 请求消息类型（Request Message Type）

RIP请求消息可以请求整个路由表信息，也可以仅请求某些具体路由的信息。就前一种情况而言，请求消息含有一个地址族标识字段为0（地址为0.0.0.0），度量值为16的单条路由，接收到这个请求的设备将通过单播方式向发出请求的地址回送它的整个路由表，并遵循一些规则如水平分隔（split horizon）和边界汇总（boundary summarization，在5.1.4小节中讲述）。

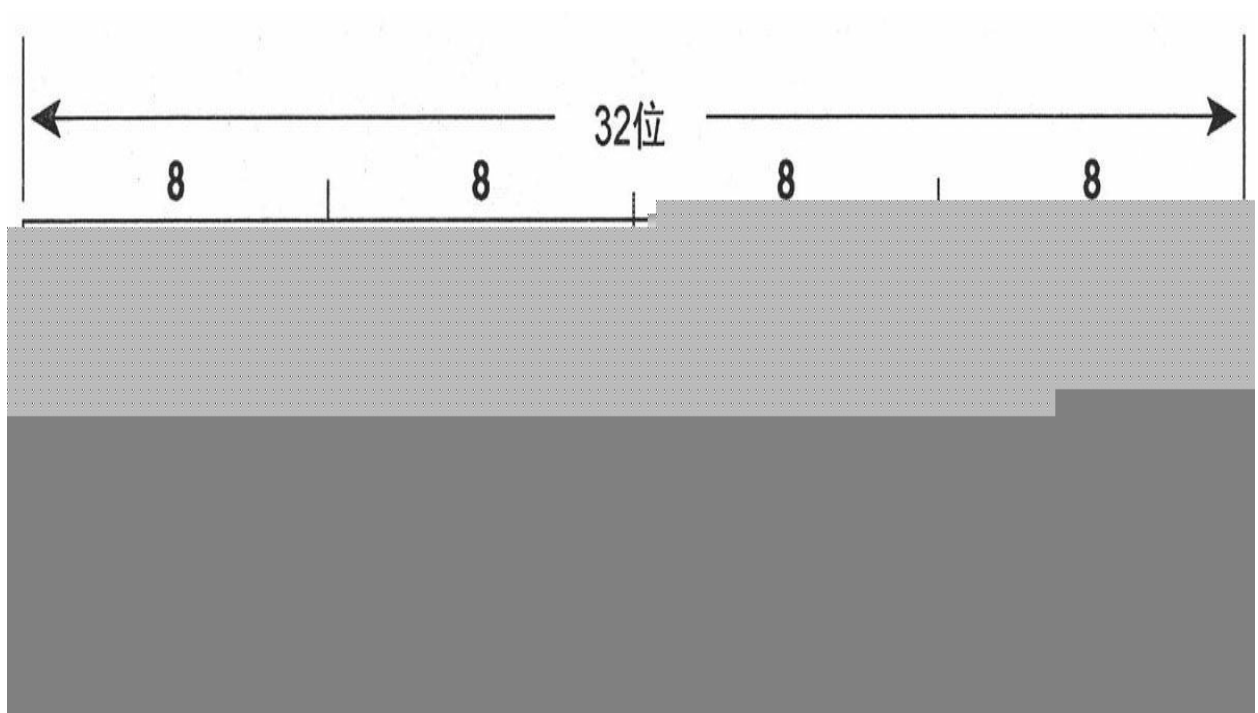
一些诊断测试过程可能需要知道某个或某些具体路由的信息。这种情况下，请求消息可以与特定地址的路由条目一起发送。接收到该请求的设备将根据请求消息逐个处理这些条目，构成一个响应消息。如果该设备的路由表中已有请求消息中地址相对应的路由条目，则将其路由条目的度量值填入metric字段。如果没有，metric字段就被设置为16。在不考虑水平分隔或边界汇总的情况下，响应消息将正确地告诉这台路由器了解的信息。

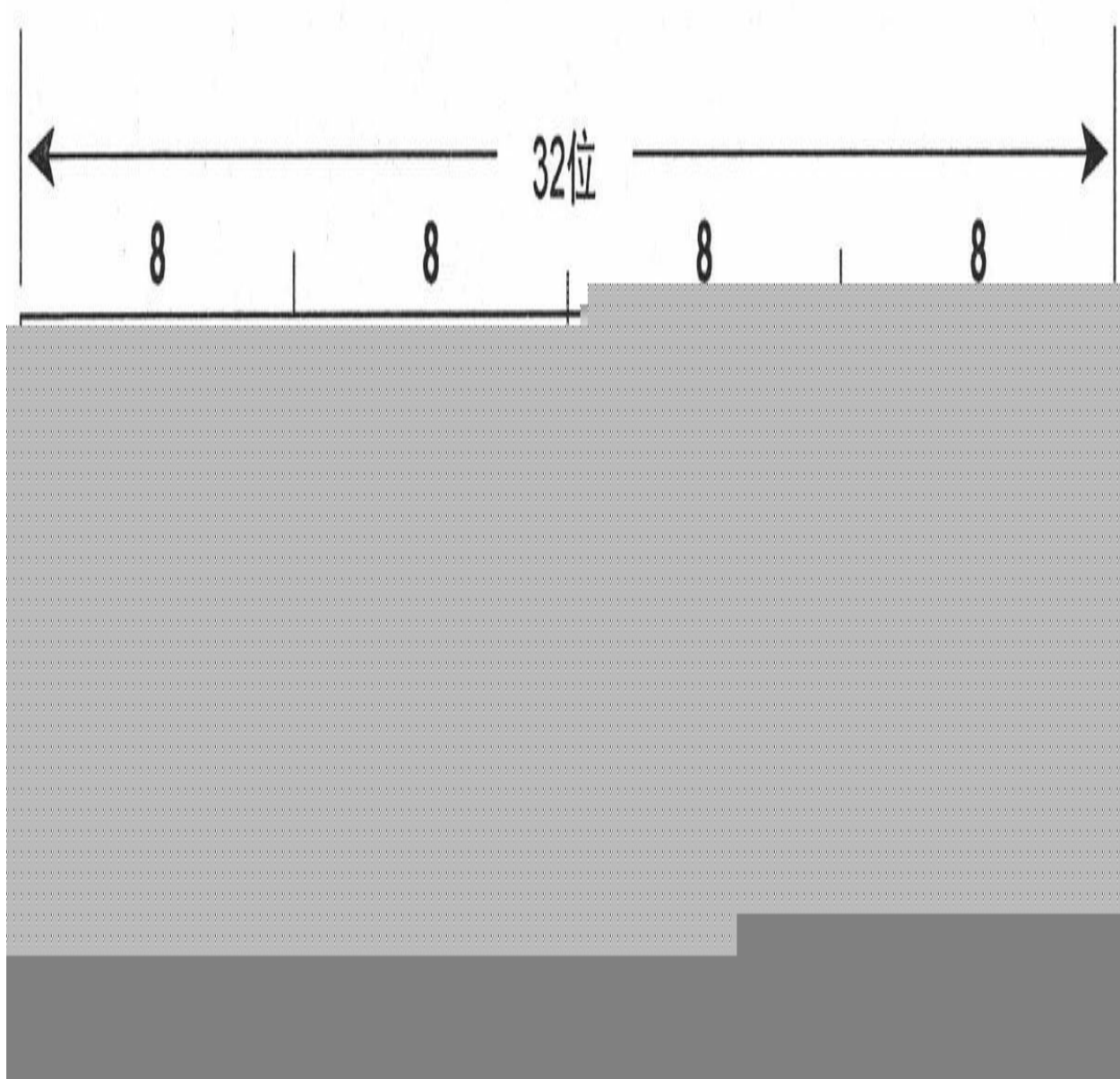
前面已经提到，主机可以在“静”模式下运行RIP。这种方法允许网络中的主机不需要发送无用的RIP响应消息，也可以通过侦听来自路由器的RIP更新来确保自己的路由表保持最新。但是，网络诊断进程可能需要检查这些“静”主机的路由表，因此，RFC 1508指出：如果一台“静”主机接收到一个来自UDP端口的请求消息，而不是来自标准的RIP 520端口，那么该主机就必须发送一个响应消息。

5.1.4 有类别路由选择（Classful Routing）

示例5-2中的路由表包含了由RIP得到的路由信息，这可以从每个路由条目左边的关键字识别出来。这些路由条目的权值由方括号中的元组来表示，与第3章中讨论的一样，第一个数字表示管理距离（administrative distance），第二个数表示度量值（metric）。从中很容易看出，RIP的管理距离为120，如前所述，RIP的度量是基于跳数的。因此，网络10.8.0.0通过E0或S1需要2跳可到达。如果到达同一个目的网络有多条跳数相等的路由，那么RIP进行等价路径的负载均衡。示例5-2中的路由表包含了多条等价路径的路由条目。

示例5-2 这个路由表包含了主网络10.0.0.0和172.25.0.0的子网，所有这些非直连网络都是从RIP协议学习得到的





当一个数据包进入宣告**RIP**的路由器后，路由器将执行路由表的查询，逐步排除数据包的路由选择范围，直到只剩下一条惟一的路径。路由器首先读出目的地址的网络部分（基于有类别路由选择协议的主网络号），查看这个网络部分在路由表中是否有其匹配的条目。根据有类别路由表查询规则的第一步，读出基于A类、B类或C类主网分类的网络号。如果没有匹配的主网络，这个数据包就被丢弃，同时发出一个**ICMP**目的不可达的消息给发出该数据包的源。如果存在匹配该数据包网络部分的主网络，那么路由表中会列出匹配这个主网络的子网，并进一步在这些子网中进行查询；此时，如果能找到一个匹配的子网条目，那么该数据包将可以被路由器转发，否则，该数据包将被丢弃并发出一

个ICMP目的不可达的消息。

1. 有类别路由选择：直连的子网络

有类别路由查询可以用下面3个例子来表述（参见示例5-2）：

（1）假设有一个目的地址为192.168.35.3的数据包进入路由器，由于该路由器在路由表中没有发现和网络192.168.35.0匹配的条目，因此该数据包将被丢弃。

（2）假设有一个目的地址为172.25.33.89数据包进入路由器，在路由表中有一个和B类网络172.25.0.0/24匹配的条目，那么进一步检查路由表中列出的网络172.25.0.0/24的子网条目；显然没有和网络172.25.33.0匹配的子网条目，因此该数据包被丢弃。

（3）最后一个例子，假设要到达地址172.25.153.220的数据包进入路由器，这时，路由表中有和网络172.25.0.0/24匹配的条目，进一步检查到有和子网172.25.153.0匹配的条目，因此，该数据包将被转发到下一跳地址172.25.15.2。

另外，请注意观察图5-2中所显示的一个情况，RIP协议的消息中并没有随同路由条目一起通告子网掩码，从而路由器中也没有和单独的子网相关联的掩码。因此，一台路由器的转发数据库如同示例5-2中所显示的那样，当它收到了一个目的地址为172.25.131.23的数据包，即使这个地址被完全子网化，路由器也没有确切的方法来识别子网位的结束位置和主机位的开始位置。

路由器惟一可以借助的就是，假定在整个网络中，对于同一个主网络地址使用相同的掩码，这个掩码就是配置在与网络172.25.0.0相连的某一台路由器接口之上的。对于从目的地址的子网部分得出的主网络172.25.0.0，它将使用自己的掩码。正如本章所有图示中所显示的路由表那样，如果一个网络是和路由器直连的，那么路由器将在路由表中作为一个标题条目列出该网络 and 该网络所连接的接口的子网掩码，然后列出它所知道的关于这个网络的所有子网。如果一个网络和路由器不是直接相连的，那么路由表将仅仅列出这个网络的主网络而不列出与它相关联的掩码。

因为在有类别路由选择协议进行路由选择的情况下，数据包的目的地址

是通过在路由器接口本地配置的子网掩码来识别的，所以在同一个主网络范围中的所有子网掩码应该是一致的。

2. 有类别路由选择：在边界路由器上的路由汇总

在前面的讨论中产生了这样一个问题，就是当一个网络没有和路由器的任何接口相连接时，RIP协议应该如何识别一个主网络的子网呢？如果没有一个接口和该目的地址对应的A类、B类或C类主网络相关联，那么路由器将没有办法正确地识别出所使用的子网掩码，也没有办法正确地标识该子网。

解决方法很简单：如果路由器没有和某个目的网络直接连接，那么该路由器仅仅需要一条简单的路由指向一个直接相连的路由器。

图5-4显示了一台处于两个主网络边界上的路由器，这两个主网络是A类网络10.0.0.0和C类网络192.168.115.0。这台“边界”路由器不会把其中一个主网络的子网的具体信息发送给另一个主网。正如所显示的那样，该路由器执行了自动路由汇总，或称为子网屏蔽（subnet hiding），仅把地址10.0.0.0通告给网络192.168.115.0，同样的会把地址192.168.115.0通告给网络10.0.0.0。

根据这种方式，在网络192.168.115.0内的路由器的路由表中，只包含一个单独的路由条目用来引导将到达目的网络10.0.0.0的数据包转发到边界路由器，而边界路由器则具有和网络10.0.0.0直连的接口，因此它具有一个带子网掩码的子网来为在网络“云”内的数据包选路。示例5-3显示了一台在网络192.168.115.0中的路由器的路由表，在路由表中可以看到有一条网络10.0.0.0的路由条目，它看起来像是一条单独的、没有子网掩码的路由条目。

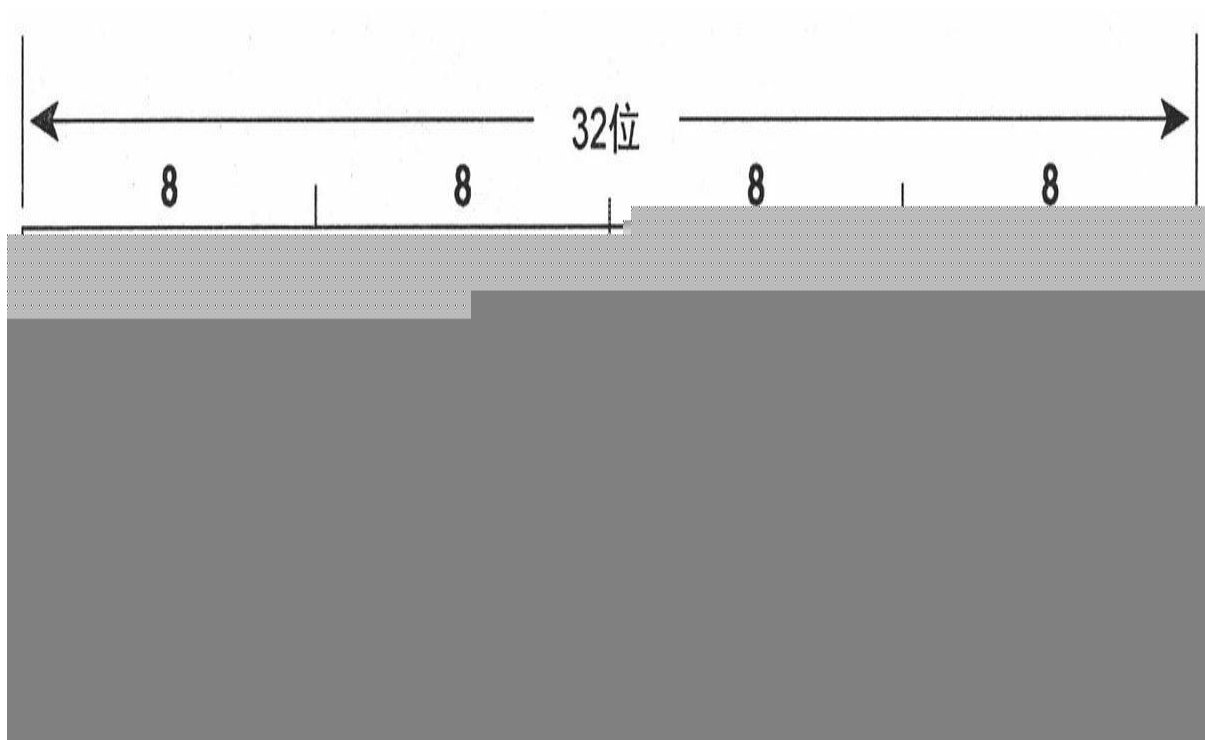
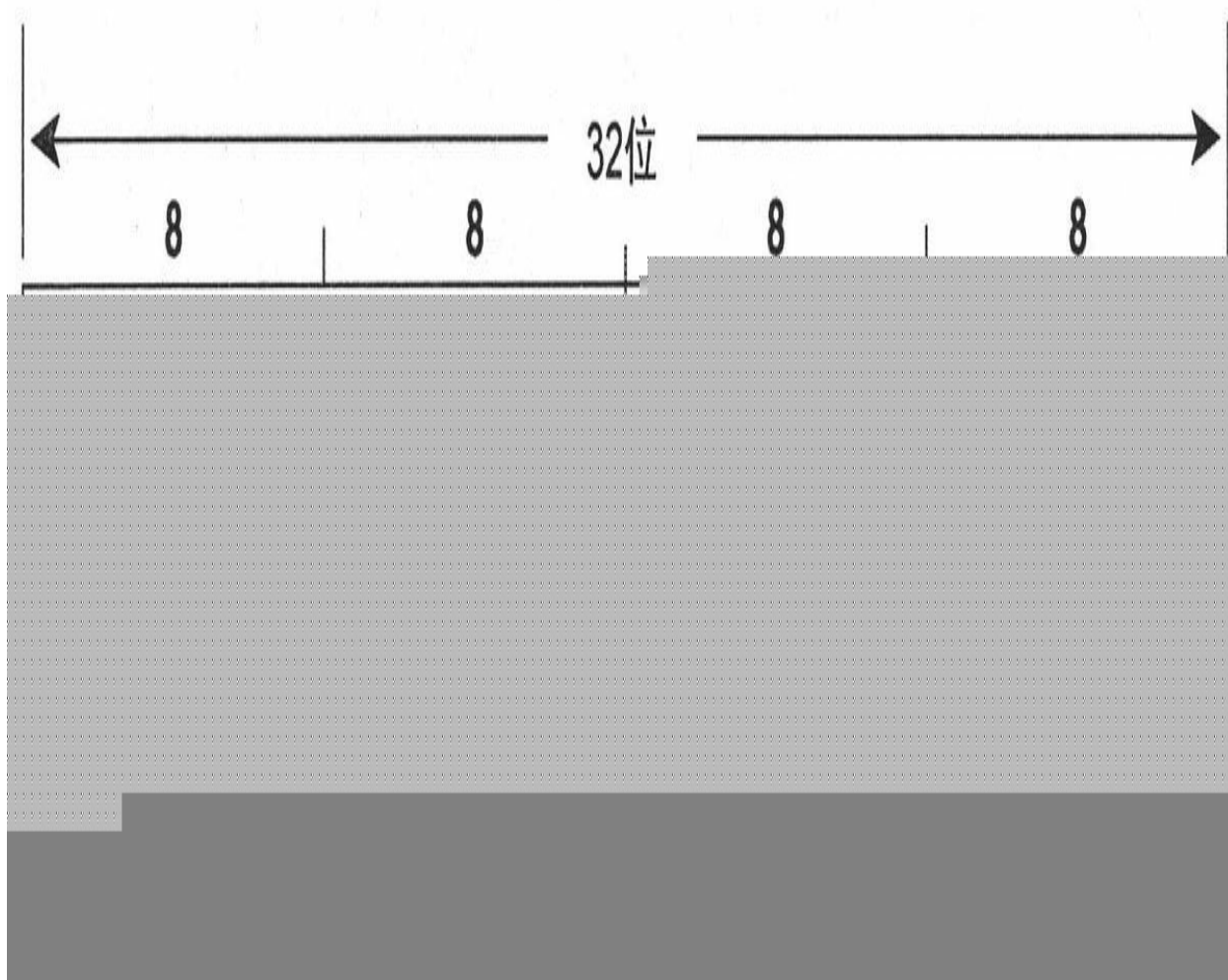


图5-4 处在两个主网络边界上的路由器不会把其中一个主网络的子网通告给另一个主网

示例5-3 这台路由器有一个单独的路由条目指向网络**10.0.0.0**，该网络可以经过**1**跳到达，所以它的下一跳地址就是边界路由器



第3章简要讨论了一个主网络被另一个不同的主网络分割成不连续的子网的情况。这里要注意，这种情况在像**RIP**、**IGRP**等有类别路由协议中会带来一些问题，不连续的子网在网络边界会被自动汇总。5.2节中对这类问题和及其解决办法进行了描述。

3. 有类别路由选择：小结

有类别路由选择协议的一个基本特征是，在通告目的地址时不能随之一一起通告它的地址掩码。因此，有类别路由选择协议首先必须匹配一个与该目的地址对应于A类、B类或C类的主网络号。对于每一个通过这台路由器的数据包：

（1）如果目的地址是一个和路由器直接相连的主网络的成员，那么该网络的路由器接口上配置的子网掩码将被用来确定目的地址的子网。因

此，在该主网络中必须自始至终地统一使用这个相同的子网掩码。

（2）如果目的地址不是一个和路由器直接相连的主网络的成员，那么路由器将尝试去匹配该目的地址对应于A类、B类或C类的主网络号。

5.2 配置RIP

与RIP协议简易的特点相对应的是，RIP协议的配置工作也是一项比较简单的事情。首先，用一条命令来启动RIP进程，另外还有一条命令用来指定运行RIP协议的每一个网络。在此之后，就只有很少的几个配置选项了。

5.2.1 案例研究：一种基本的RIP配置

配置一个RIP协议，只需两步：

步骤1： 使用**router rip** 命令启动RIP进程。

步骤2： 使用**network** 命令指定每一个需要运行RIP协议的主网络。

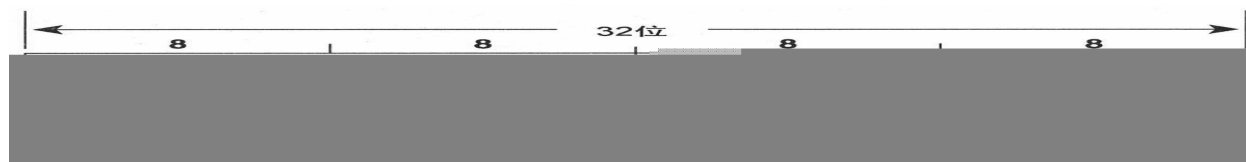
图5-5显示了一个含有4台路由器的网络，它包含4个主网络号。路由器Goober和网络172.17.0.0的两个子网相连。



图5-5 在按照主网络分类的层面上，路由器Andy和Barney都是网络之间的边界路由器

在示例5-4中显示了启动RIP协议所必需的命令。

示例5-4 路由器Goober的RIP配置



类似的，路由器Opie和同一个网络172.17.0.0的两个子网相连，相应的配置命令参见示例5-5。

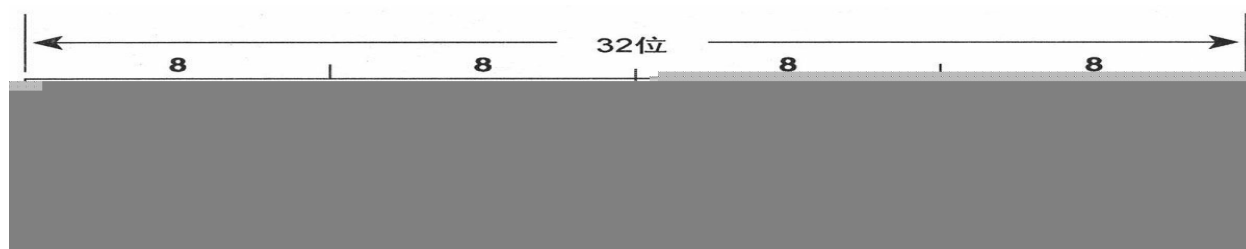
示例5-5 路由器Opie的RIP配置



使用任何一个**router** 命令都需要让路由器进入到**config-router** 配置模式，这可以通过提示符辨别出来。由于RIP协议具有有类别路由选择的特性，从而在网络边界上会出现子网屏蔽的情形，这意味着**network** 命令中不需要指定子网，而仅仅需要指定相对应的A类、B类或C类的主网络地址。任何一个接口，只要它的配置地址属于**network** 命令指定的网络，都将会运行RIP。

路由器Barney和两个主网络——10.0.0.0和192.168.83.0相连。因此，这两个主网络都需要被指定，参见示例5-6。

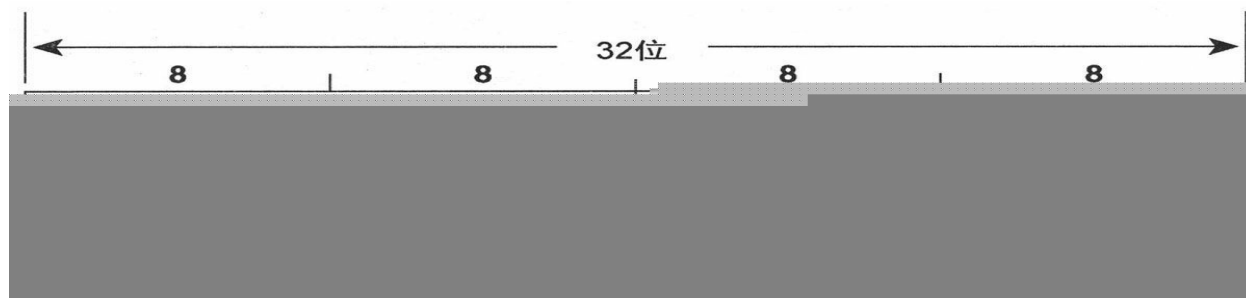
示例5-6 路由器Barney的RIP配置



路由器Andy和网络192.168.83.0的一个子网相连，和网络192.168.12.0的两个子网相连，和网络172.17.0.0的两个子网相连。示例5-7显示了它们

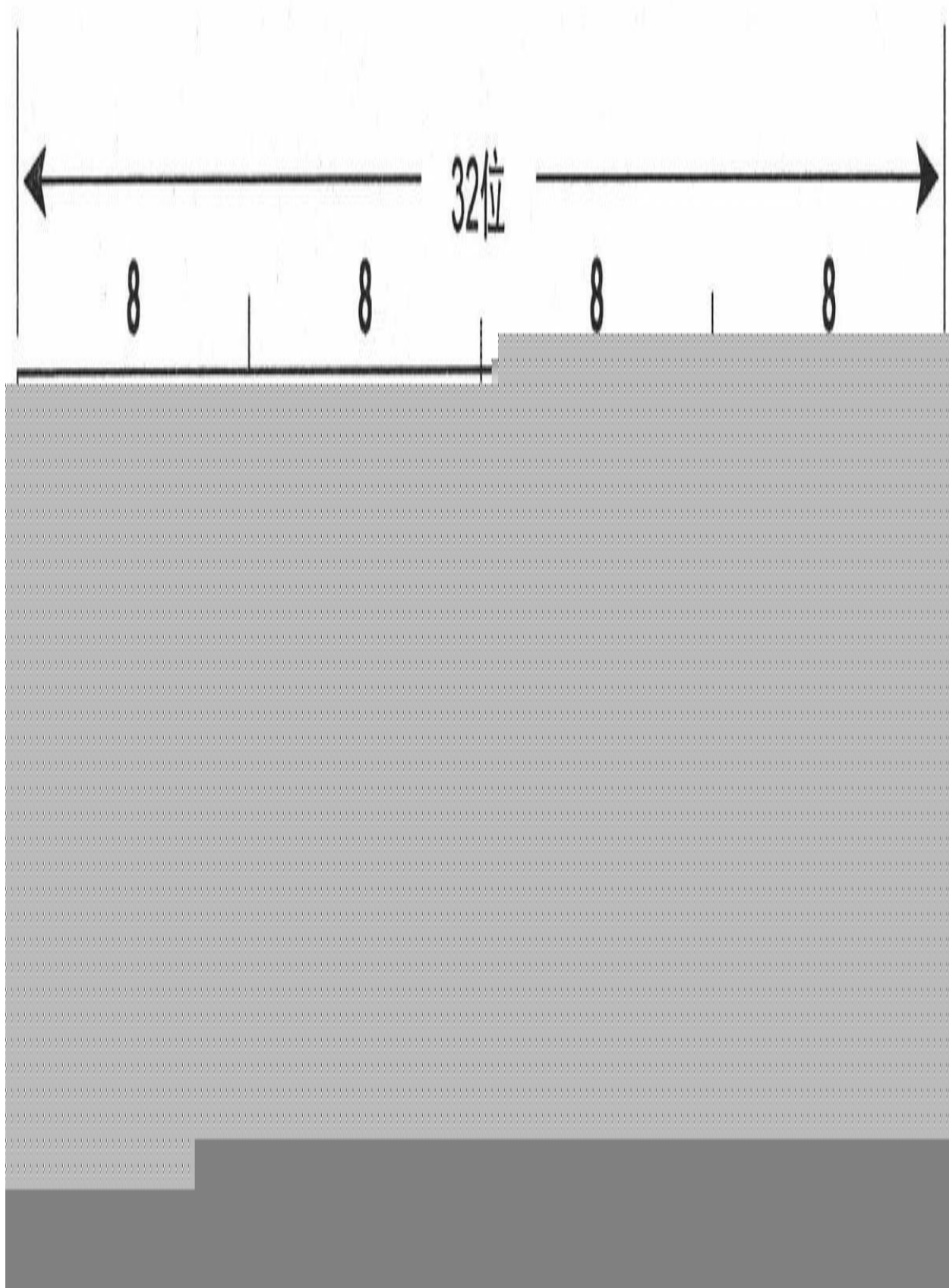
的配置。

示例5-7 路由器Andy的RIP配置



在示例5-8中，在路由器Andy上打开了RIP协议的调试命令**debug ip rip**。这里要特别注意的是，路由器Andy执行了子网屏蔽。由于E0和E2接口都是和网络192.168.12.0相连的，因而子网192.168.12.64和192.168.12.192可以在这两个接口上进行通告；但是在和其他不同的网络相连的E1、S0和S1接口上，这两个子网则被进行路由汇总后才通告出去。同样的，网络192.168.83.0和网络172.17.0.0在通过有类别网络边界时也会被路由汇总后通告出去。注意，路由器Andy正在接收一条来自路由器Barney关于网络10.0.0.0的汇总路由。最后，从示例中也可以观察到水平分隔的情况。例如，从E1接口通告给路由器Barney的路由中不再包含网络10.0.0.0或192.168.83.0的路由条目。

示例5-8 这些**debug**消息显示了路由器Andy上接收和发送的**RIP**更新消息，并可以观察到网络路由的汇总和水平分隔的效果



5.2.2 案例研究：被动接口（Passive Interface）

在图5-6所示的网络中增加路由器Floyd，但不希望在路由器Floyd和Andy之间交换RIP协议的通告消息，这在路由器Floyd上可以很容易地实现，参见示例5-9。

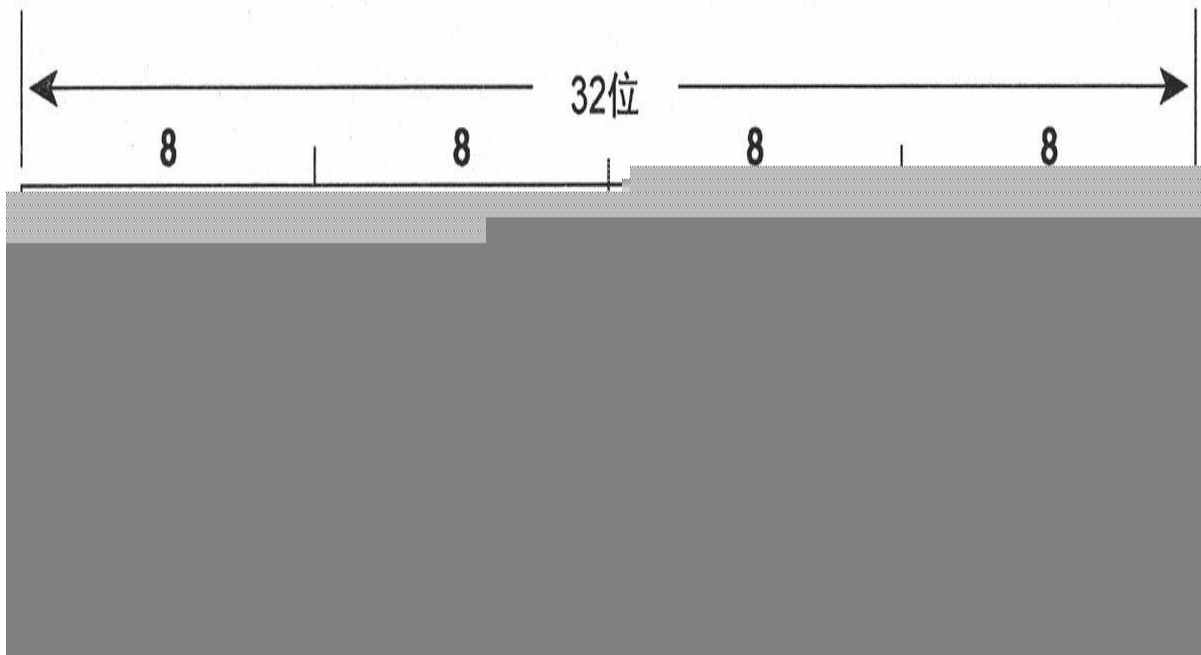
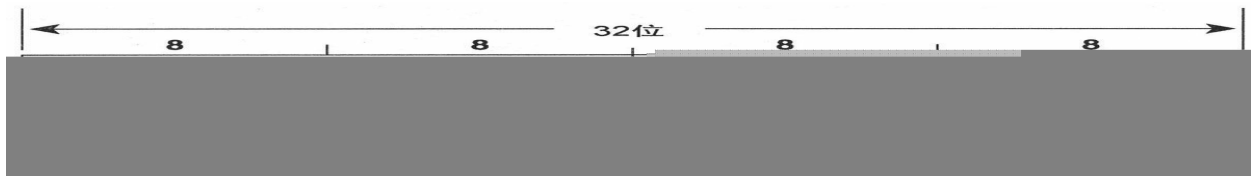


图5-6 网络的路由策略要求在路由器Andy和Floyd之间没有RIP消息的交换

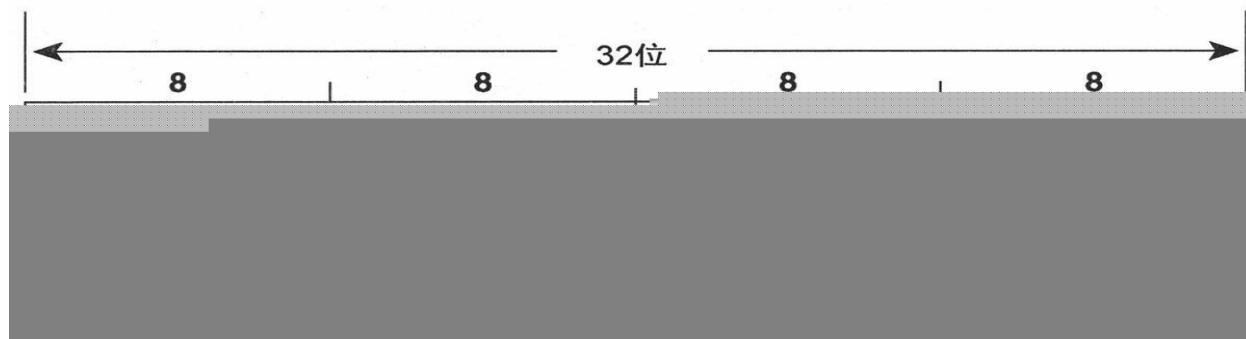
示例5-9 路由器Floyd的RIP配置



由于没有包含网络192.168.12.0的**network** 命令语句，路由器Floyd将不在接口192.168.12.66上通告192.168.12.0。但是，路由器Andy有两个和网络192.168.12.0相连的接口，因此网络192.168.12.0必须包含在RIP中。如果一台路由器接口处于一个启动RIP协议的网络的子网中，那么路由器会在该接口上发出RIP广播，为了阻塞这样的RIP广播，在RIP的处理

中就需要增加一条**passive-interface** 命令。路由器Andy的RIP配置参见示例5-10。

示例5-10 在路由器Andy上，具有被动接口的RIP配置



命令**Passive-interface** 不是**RIP** 协议专有的命令，它可以在所有的IP路由选择协议中配置使用。使用命令**passive-interface** 实际上可以说是在一条特定的数据链路上，将路由器作为一台“静”主机来看待。像其他的“静”主机一样，它只是在该特定的链路上侦听RIP的广播，从而更新自己的路由表。如果希望避免路由器从一条链路上学到路由信息，就必须使用更复杂的路由更新控制才能实现，这种路由更新控制称为出站更新过滤（**filtering out updates**）（路由过滤的内容将在第13章中讨论）。和“静”主机不同的是，路由器并不在被动接口上响应收到的请求消息。

5.2.3 案例研究：配置单播更新（Unicast Update）

接下来，增加一台新的路由器Bea，并连接到路由器Andy和Floyd之间的以太网共享链路上（如图5-7所示）。在路由器Andy和Floyd之间的链路上依然保留不启动RIP协议的路由策略，但在路由器Bea和Andy之间，Bea和Floyd之间现在都必须交换RIP通告消息。

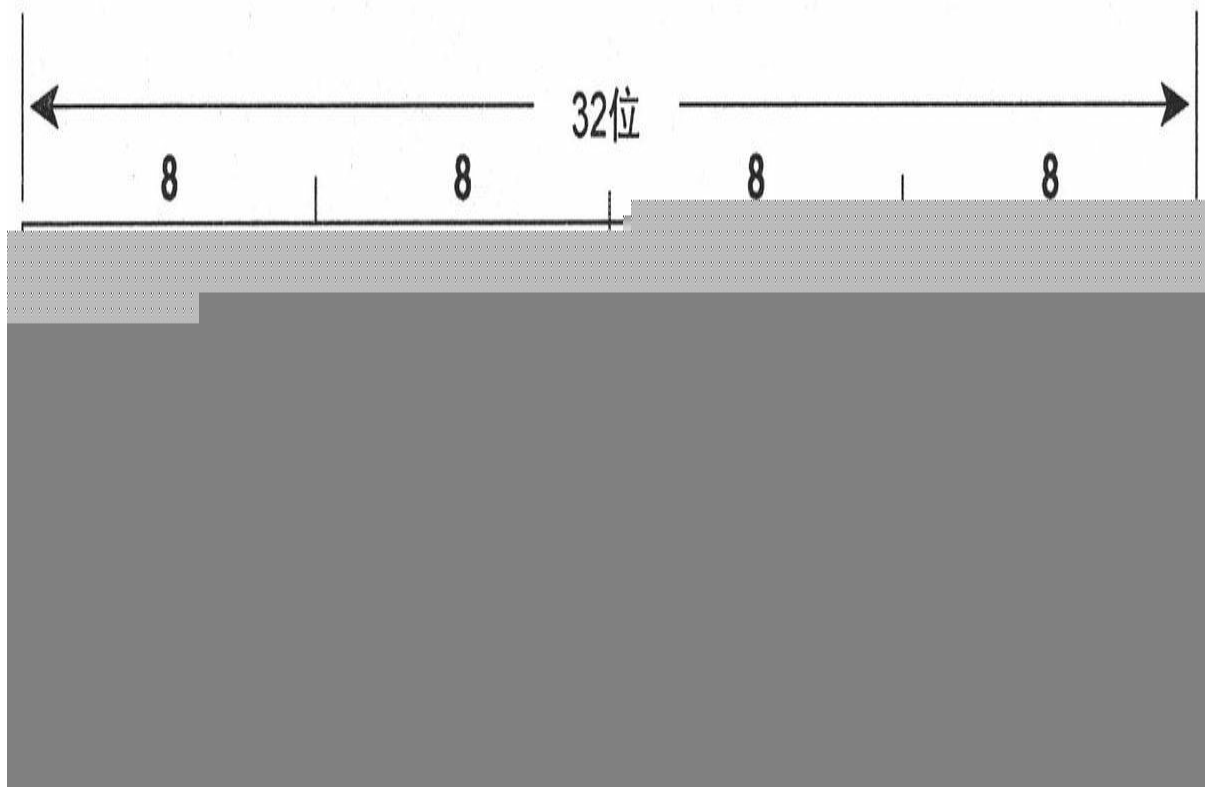
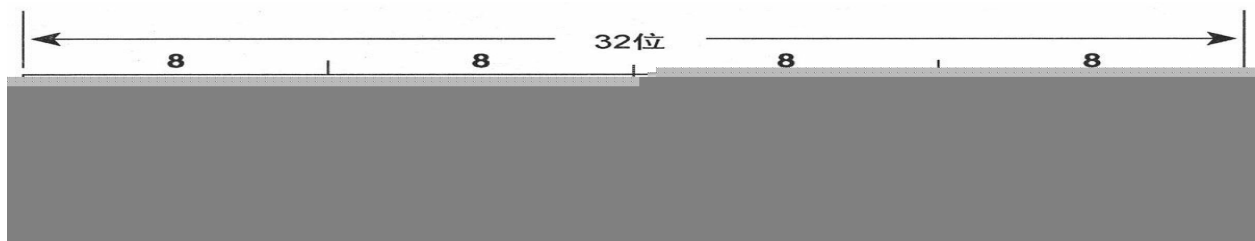


图5-7 路由器Andy和Floyd不进行RIP更新的交换，但是路由器Andy和Floyd都与路由器Bea交换RIP更新

路由器**Bea** 的配置很简单，参见示例5-11。

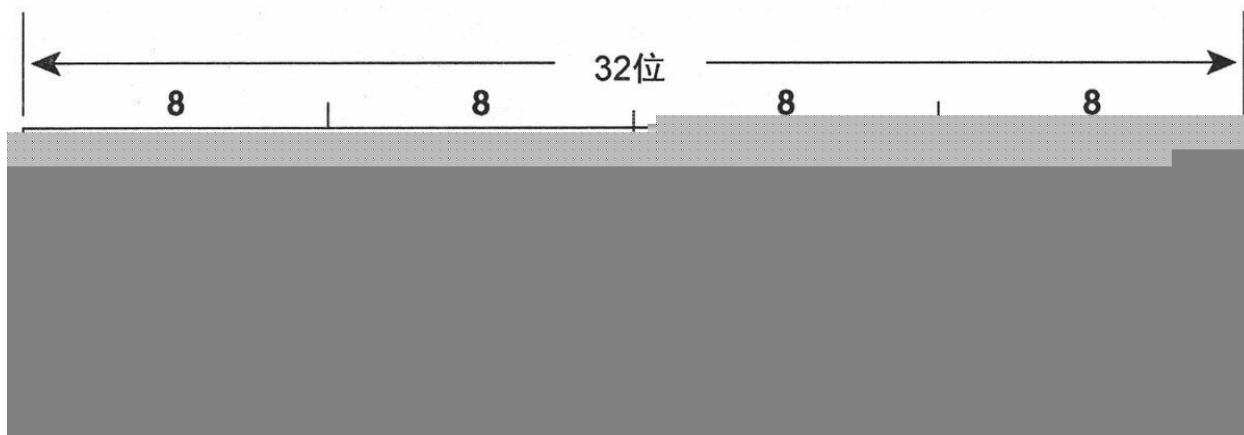
示例5-11 路由器**Bea**的RIP配置



在路由器Andy的RIP处理中增加一条额外的命令**neighbor**，使RIP协议能够以单播方式向路由器Bea的接口发送通告，而这时，路由器Andy上的**passive-interface** 命令仍继续防止在该链路上广播更新。[\[9\]](#)

路由器Andy的配置参见示例5-12。

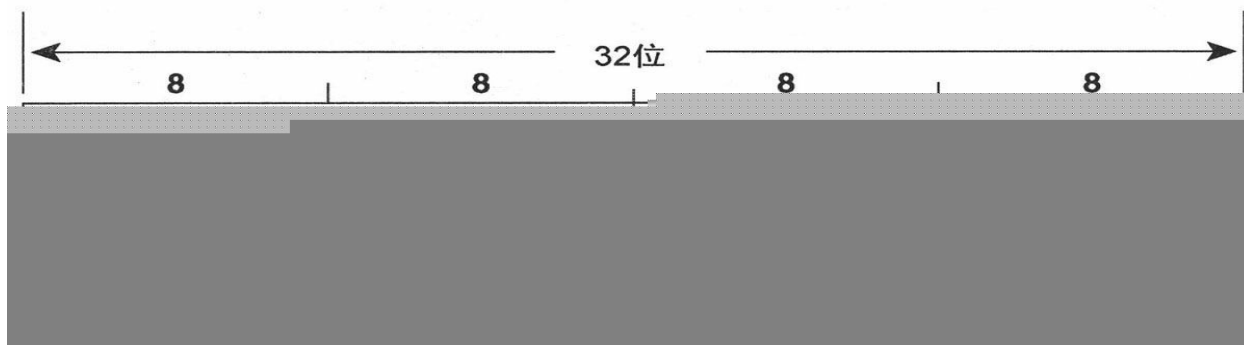
示例5-12 路由器Andy的RIP配置，其中包括带有启动单播更新的neighbor语句的被动接口



因为路由器Floyd现在必须发送RIP通告给路由器Bea，因此也必须增加一条通告 192.168.12.0的network命令。为了防止广播更新，也要增加一条Passive-interface命令，并

且要增加neighbor命令以单播方式通告RIP更新给路由器Bea，参见示例5-13。

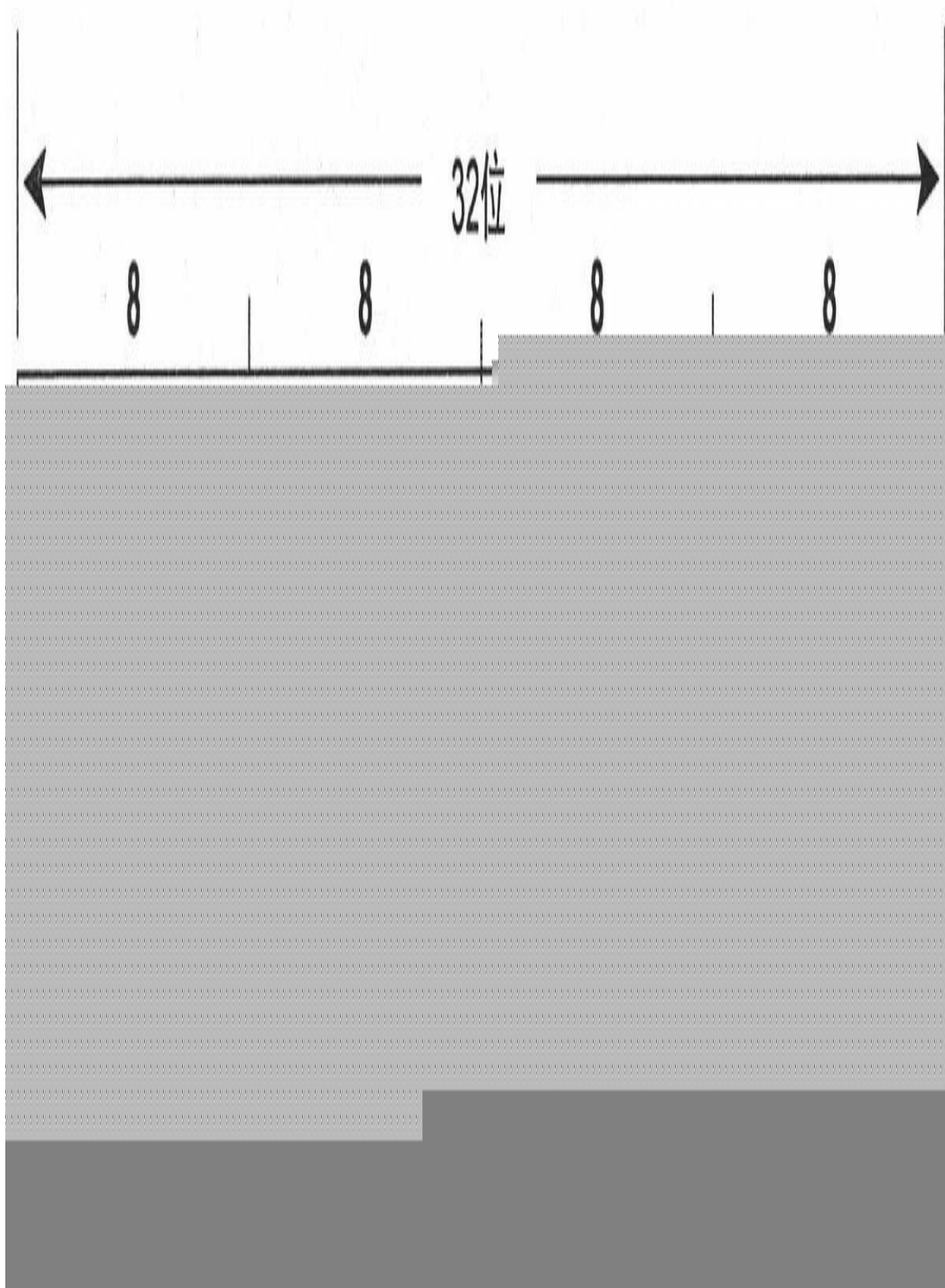
示例5-13 路由器Floyd使用neighbor 192.168.12.67配置为单播更新



在路由器Andy上启动调试命令debug ip rip events，可以用来验证更改后的新配置的效果（参见示例5-14）。路由器Andy可以从路由器Bea收到路由更新，但是无法从Floyd上收到，并且正在以单播方式直接给路由器Bea的接口发送路由更新，但是却不在它的E0接口上进行广播。

示例5-14 路由器Andy在E0接口发送出的惟一路由更新是到路由器Bea的单播更新。路由器Andy可以收到来自路由器Bea的更新，而不是

路由器**Floyd**的更新



虽然路由器Bea可以从路由器Andy和路由器Floyd学到路由，而且在共享的以太网上广播更新，但由于水平分隔法则依然适用，因此可以防止路由器Bea将从那两台路由器学到的路由重新通告到它们之间的以太网上。

5.2.4 案例研究：不连续的子网

在图5-8中，另一台路由器被添加到原来的网络上，它通过一个E1接口和子网10.33.32.0/20 相连。现在问题出现了，网络10.0.0.0的另一个子网——子网10.33.0.0/20是和路由器Barney相连的，它和子网10.33.32.0/20之间只有一条惟一的经过网络192.168.83.0和192.168.12.0的路由路径，而这是它们完全不同的两个网络。结果，网络10.0.0.0将变成为不连续的。

路由器Barney会认为自己是网络10.0.0.0和网络192.168.83.0之间的边界路由器；同样的，路由器Ernest_T也会认为自己是网络10.0.0.0和网络192.168.12.0之间的边界路由器。它们都将通告一条网络10.0.0.0的汇总路由，结果路由器Andy将会“傻乎乎”地认为它有两条等价的路径可以到达同一个网络。在这种情况下，路由器Andy将在与路由器Barney和Ernest_T相连的链路上进行均分负载，因而，要到达网络10.0.0.0的数据包现在只有50%的机会可以转发到正确的子网上。

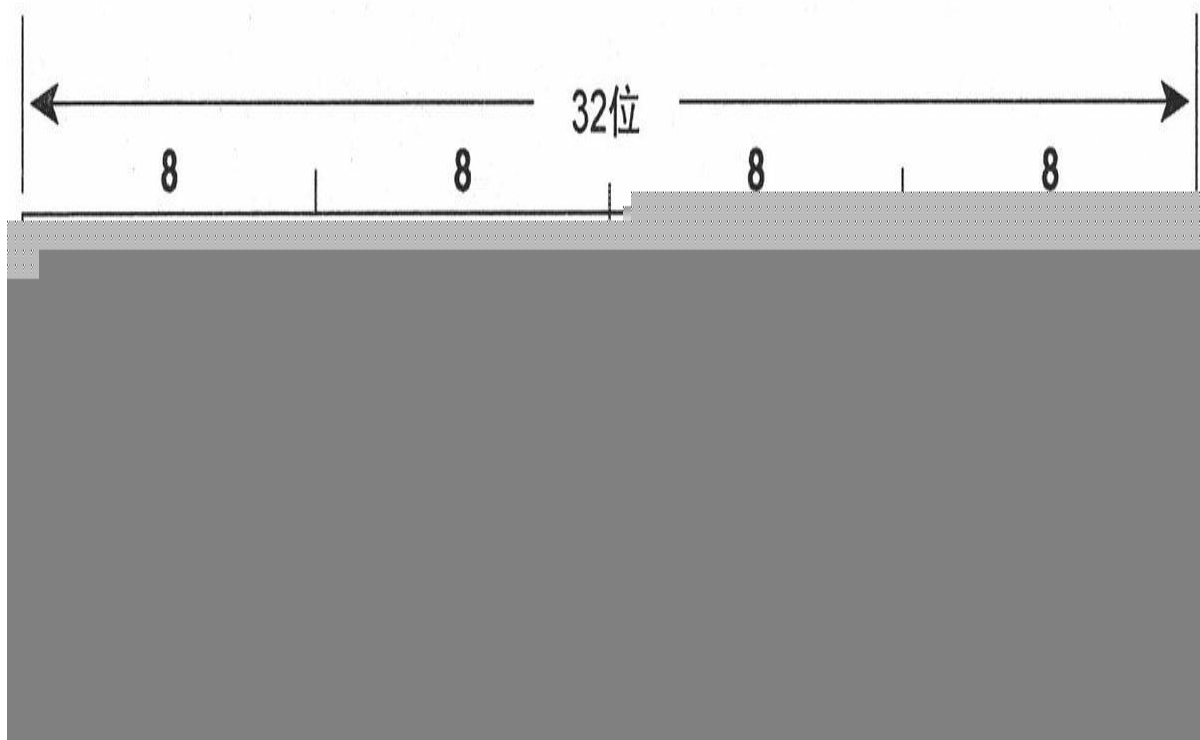


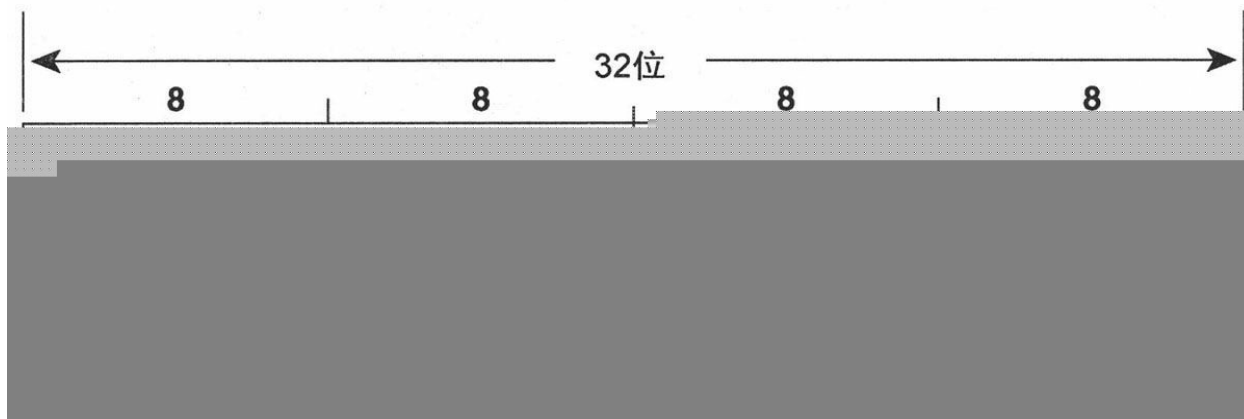
图5-8 像RIP和IGRP等有类别路由选择协议不能对图中所示类型的网络拓扑进行路由选择，因为其中的网络10.0.0.0的子网被不同的主网络分开了

解决方法是在网络192.168.83.0/24和192.168.12.192/27所在的同一条链路上配置网络10.0.0.0的子网，这可以通过在路由器接口上配置辅助IP地址（secondary IP address）实现，所有配置参见示例5-15、示例5-16和示例5-17。

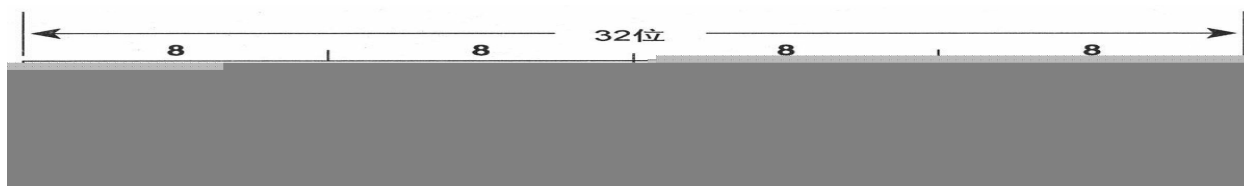
示例5-15 路由器**Barney**配置了辅助IP地址



示例5-16 路由器**Andy**配置了辅助IP地址，并在**RIP**中增加了一个新的网络



示例5-17 路由器Ernest_T配置了辅助IP地址



因为路由器Andy在前面的配置中没有和网络10.0.0.0相连的接口，所以在RIP配置中增加了一条网络声明（**network 10.0.0.0**）。配置的效果可以从图5-9中看到，原有的逻辑网络结构依然保留，只是在其网络结构上“叠加（overlaid）”了一个连续的网络10.0.0.0。

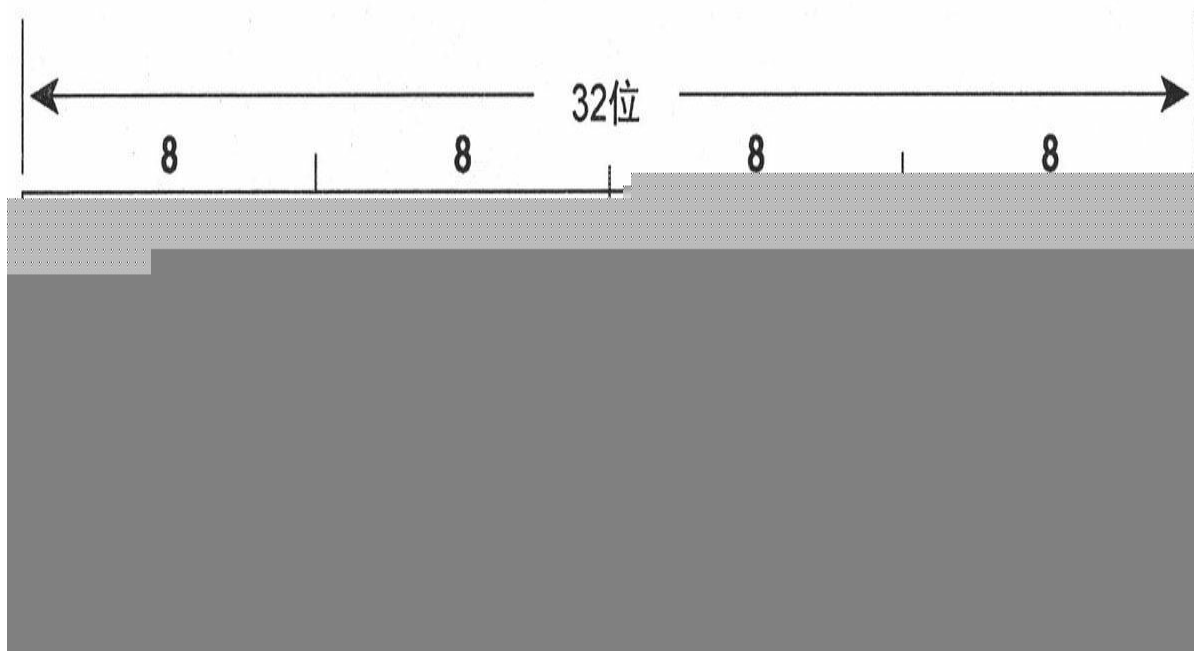


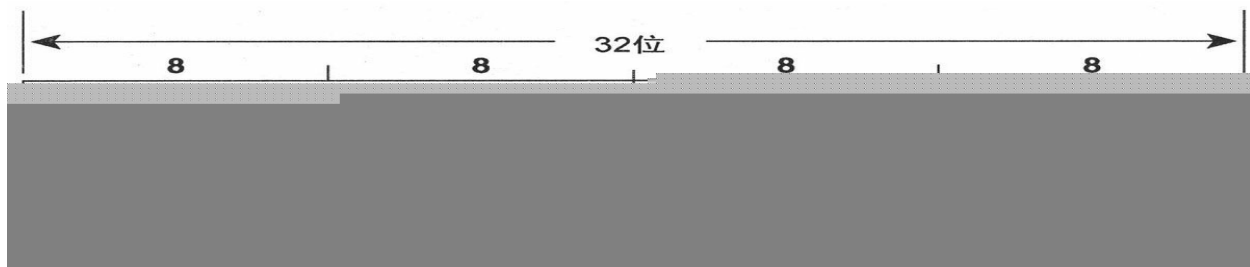
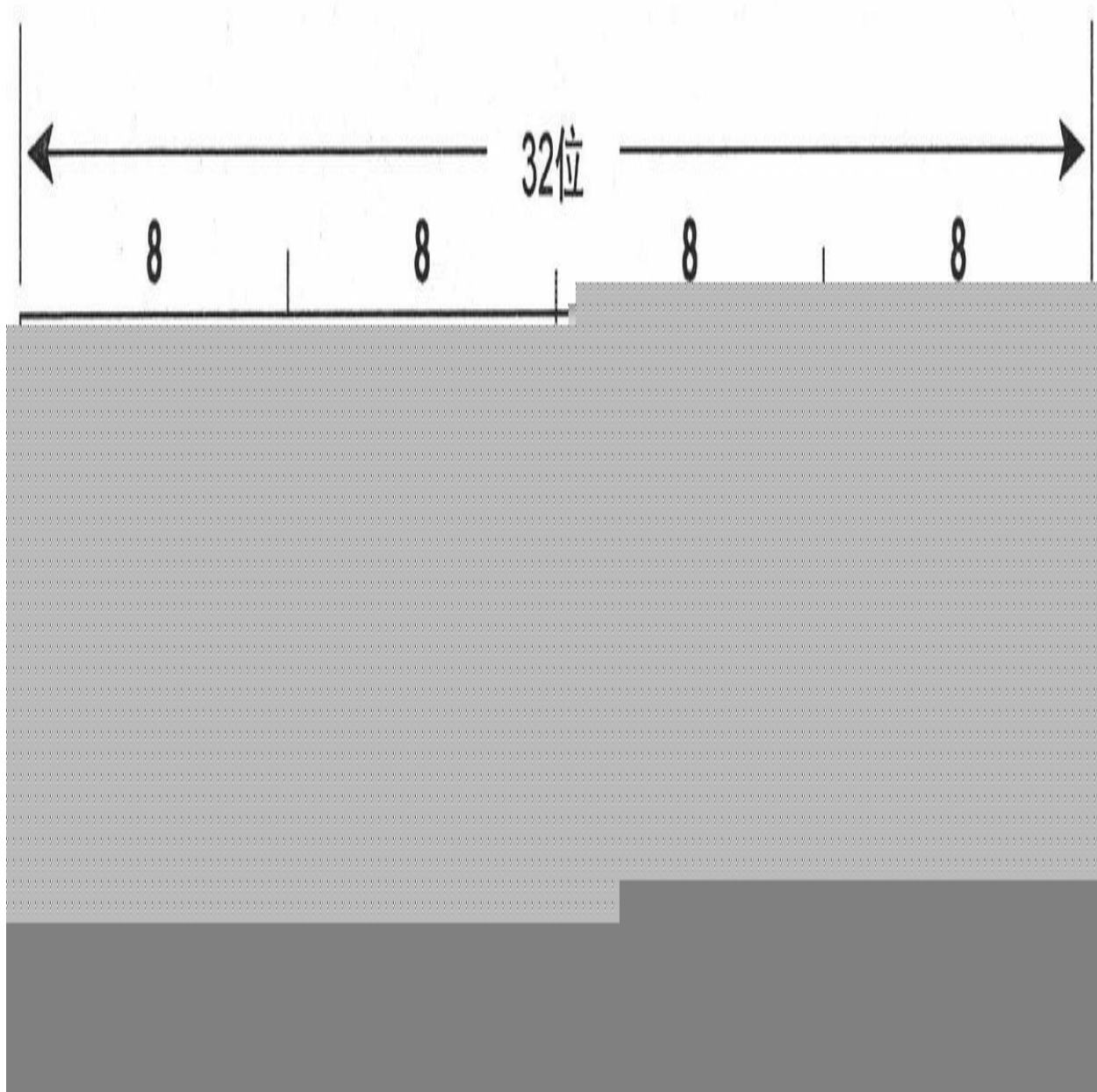
图5-9 辅助地址被用来在已有其他网络地址的同一条链路上连接网络10.0.0.0的子网

示例5-18显示了路由器Ernest_T的路由表。这里值得注意的是，两条等价路由分别和下一跳地址10.33.75.1及192.168.12.195相关联。

由于路由选择进程会将辅助地址看作是单独的数据链路，所以在RIP协议或IGRP协议网络的设计中要很小心地使用。各自的RIP更新会在每一个子网里进行广播，如果路由更新比较多而且物理链路的带宽有限（例如串行链路），那么大量的路由更新会造成网络的拥塞。在本章后面的示例5-21中，可以看到在这种配置了辅助地址的链路上会产生大量的路由更新。

在插入辅助地址时一定要格外小心，如果不小心忽略了关键字**secondary**，路由器将会认为该接口的主地址被一个新的地址代替了。在一个正在提供服务的网络接口上犯这种错误将会带来严重的后果。

示例5-18 在路由器的路由选择进程中可以看到，子网**192.168.12.192/27**和**10.33.64.0/20**虽然属于同一个物理接口，但是它们却有各自的链路



[5.2.5 案例研究：控制RIP的度量](#)

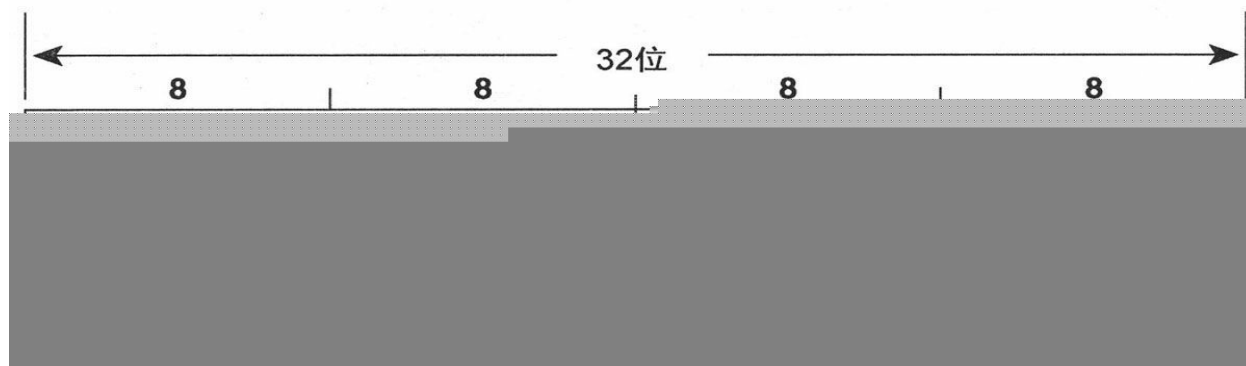
如图5-10所示，在路由器Ernest_T和Barney之间增加一条串行链路用作备份链路，只有当路由经过路由器Andy失败时这条链路才被使用。现在的问题在于，路由器Barney的子网10.33.0.0和路由器Ernest_T的子网10.33.32.0之间经过这条串行链路的路径是1跳，而经优选的以太链路的路径却是2跳。在正常的情况下，RIP会首先选择串行链路。

可以通过命令**offset-list**来改变路由的度量值，该命令指定一个数值来加大路由的度量值，并且参照一个访问列表（access list）[\[10\]](#)来决定哪些路由条目需要修改。命令语法如下所示：



路由器Ernest_T的配置参见示例5-19。

示例5-19 路由器Ernest_T在RIP配置中配置了入站偏移列表



访问列表的配置确定了关于子网10.33.0.0的路由，偏移列表（offset list）的语法含义是：“先检查从SO接口接收进来的RIP通告，如果存在和访问列表1指定的地址相匹配的路由条目，那么就把该路由条目的度量值加大2跳。”

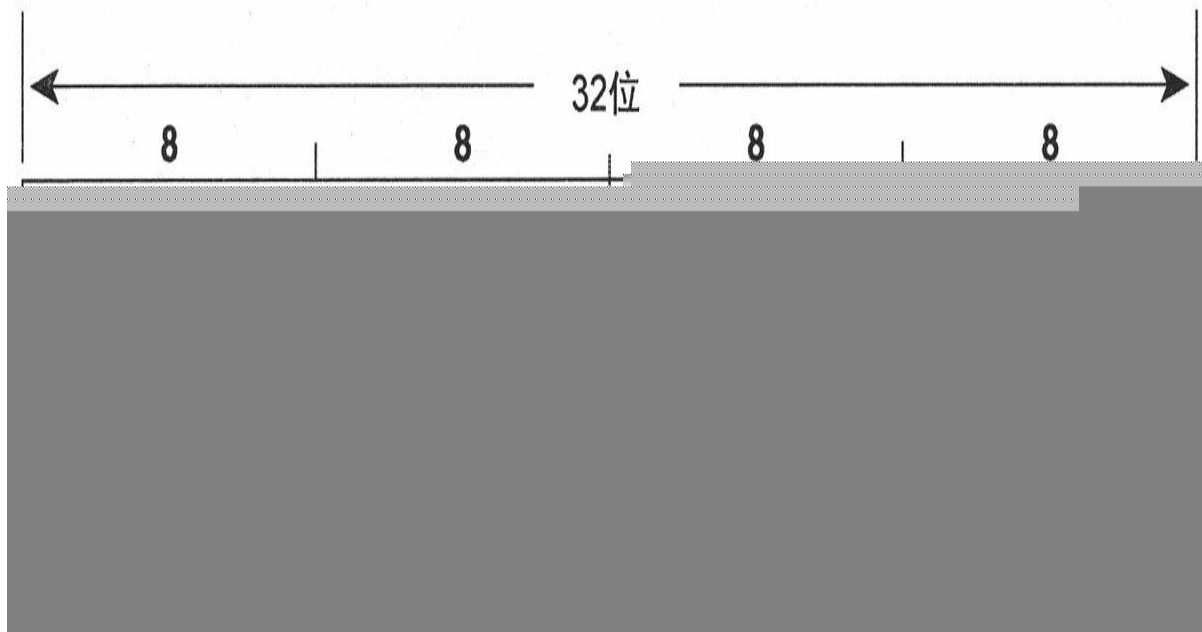


图5-10 RIP的度量值必须修改，以便使路由器Barney和Ernest_T之间2跳的以太网路由优先于1跳的串行链路路由

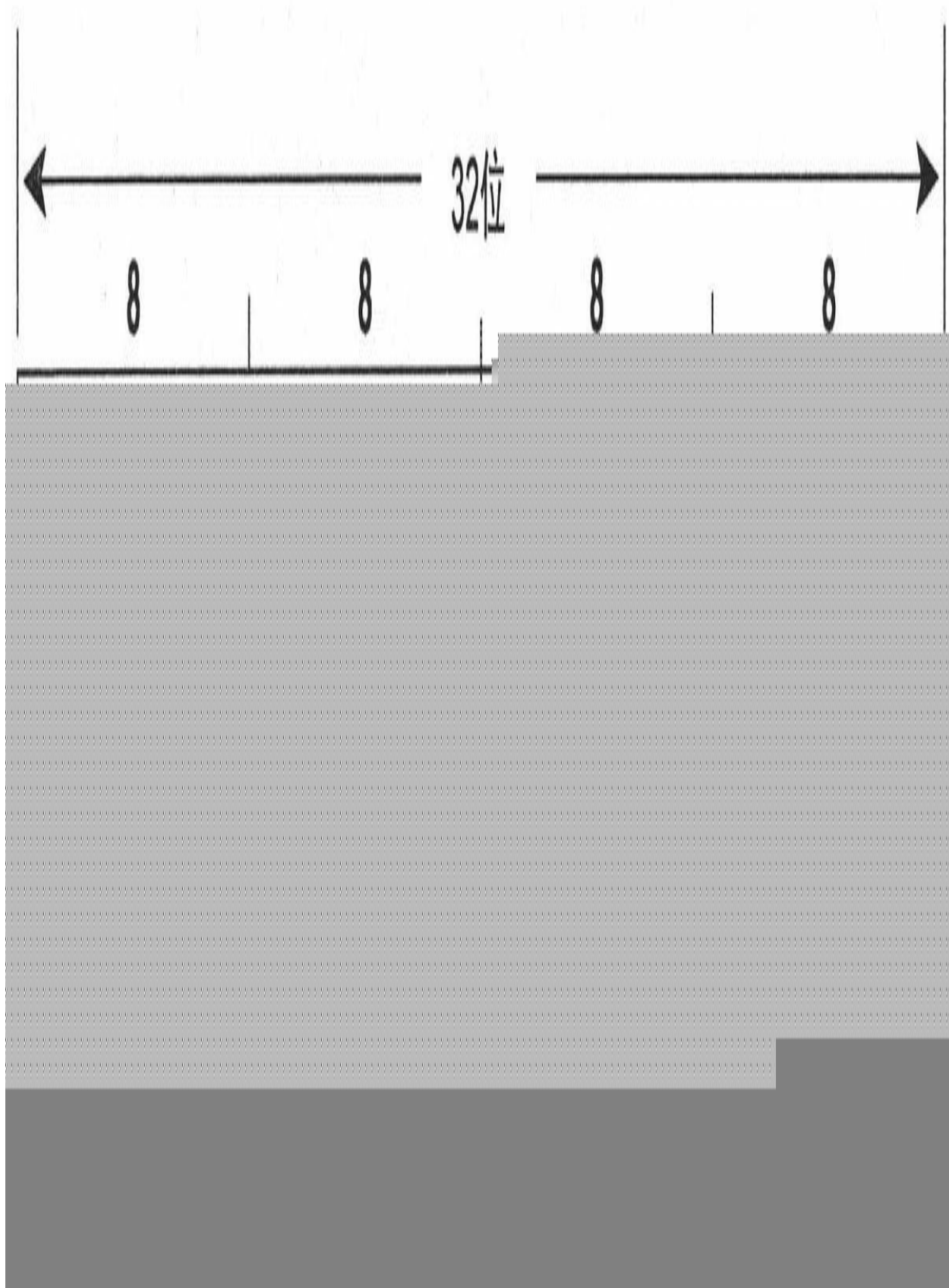
路由器Barney更新配置后，在它的配置文件中包含了示例5-20中显示的语句。

示例5-20 路由器Barney包含入站偏移列表的RIP配置



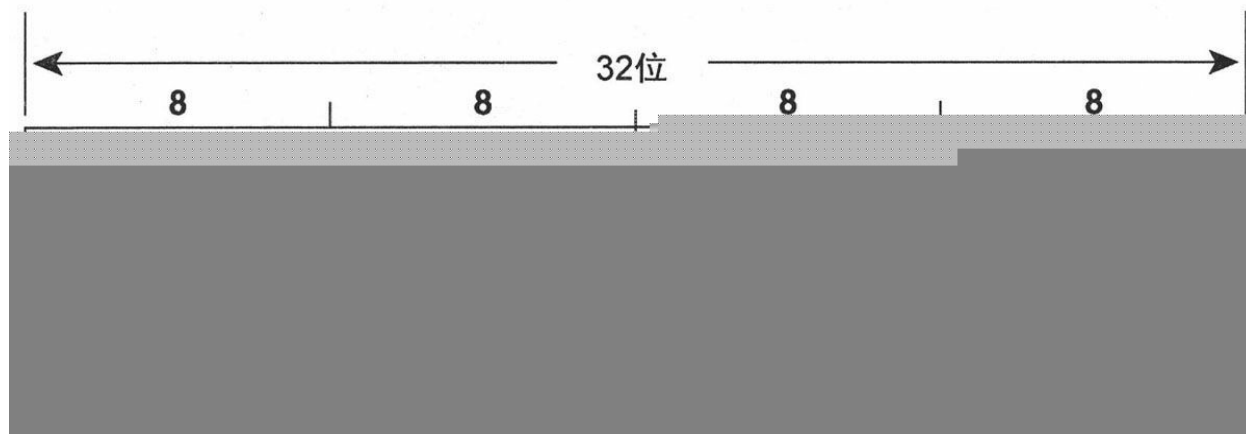
在示例5-21中显示了路由器Ernest_T的配置所产生的结果。

示例5-21 偏移列表指定额外增加的跳数，把子网10.33.0.0/20经过SO接口的路由度量由1跳变成了3跳。但经过E0接口的路由的跳数没有改变，还是2跳

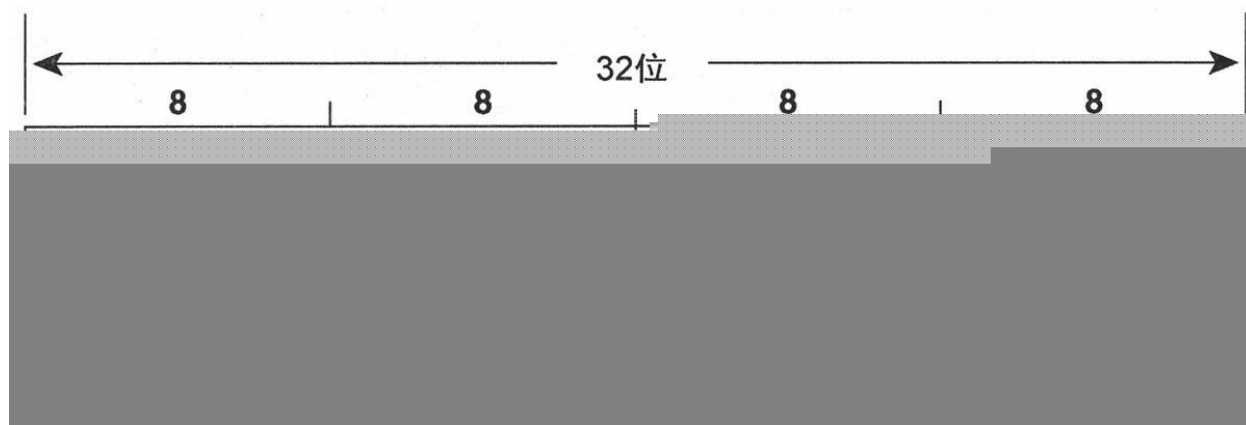


作为另一种选择，路由器也可以通过配置去修改向外通告的出站（outgoing）路由更新的度量，替代上述两台路由器对从链路上接收的入站（incoming）路由更新的度量的修改。示例5-22和示例5-23的配置可以和前面的配置达到同样的效果。

示例5-22 路由器**Ernest_T**使用出站偏移列表的配置



示例5-23 路由器**Barney**使用出站偏移列表的配置



偏移列表的其他几个选项在配置时也是有用的。如果不指定使用偏移列表的接口，那么偏移列表将在所有与访问列表匹配的接口上修改所有的入站更新或出站更新。如果不调用访问列表（使用0作为访问列表的序列号）来进行匹配，偏移列表将修改所有的入站更新或出站更新。

当在入站或出站更新的通告上选择是否使用偏移列表时，有些需要注意的地方。例如，在一个多于两台路由器的广播网络上，到底是需要某台

单独的路由器向它所有的邻居路由器广播偏移修改后的通告，还是需要某台单独的路由器接收偏移修改后的通告，这一点必须要考虑清楚。

在运行的路由上实施偏移列表时也需要特别注意。当一个偏移列表引起下一跳路由器通告的度量值比它正在通告的路由更新的度量值更高时，直到抑制计时器（holddown timer）超时前，这条路由都会被标记为不可到达。

5.2.6 案例研究：最小化更新信息的影响

如果读者希望将由路由选择更新引起的网络流量减少到最小，那么现在可以做到。在缺省情况下，普通的RIP更新消息是每30s产生一次，包括完整的路由表，除非是在路由条目的度量发生变化时传送的瞬时更新（flash update）。在具有很多子网的网络中，特别是对于一些较低带宽的链路，路由选择更新会对网络流量产生很大影响。对于正在使用流量计费方式付费的链路来说，读者也可能希望将其路由选择更新流量降低到最小。

如图5-10所示，假定从路由器Barney到Ernest_T的新的串行链路利用率很高。我们可以通过两种方式使路由选择协议产生的流量最小化，从而降低RIP的路由选择流量：第一种方法是调整路由选择协议的计时器以便降低更新的频率，但是这在主要链路发生故障时会引起较长的收敛时间；另外一种方法是配置触发扩展特性来消除周期性的RIP更新。

使用接口模式下的命令**ip rip triggered**可以启动RIP协议的触发扩展特性。[\[11\]](#)在一条串行链路上的两台路由器都必须确定配置了具有触发扩展特性的RIP协议后，路由表的更新将会变得最少，仅仅包括路由表最初的交换信息和路由表发生变化时的更新信息。这条命令仅仅在串行链路上有效，并且必须在链路的两端同时配置才会产生效果。

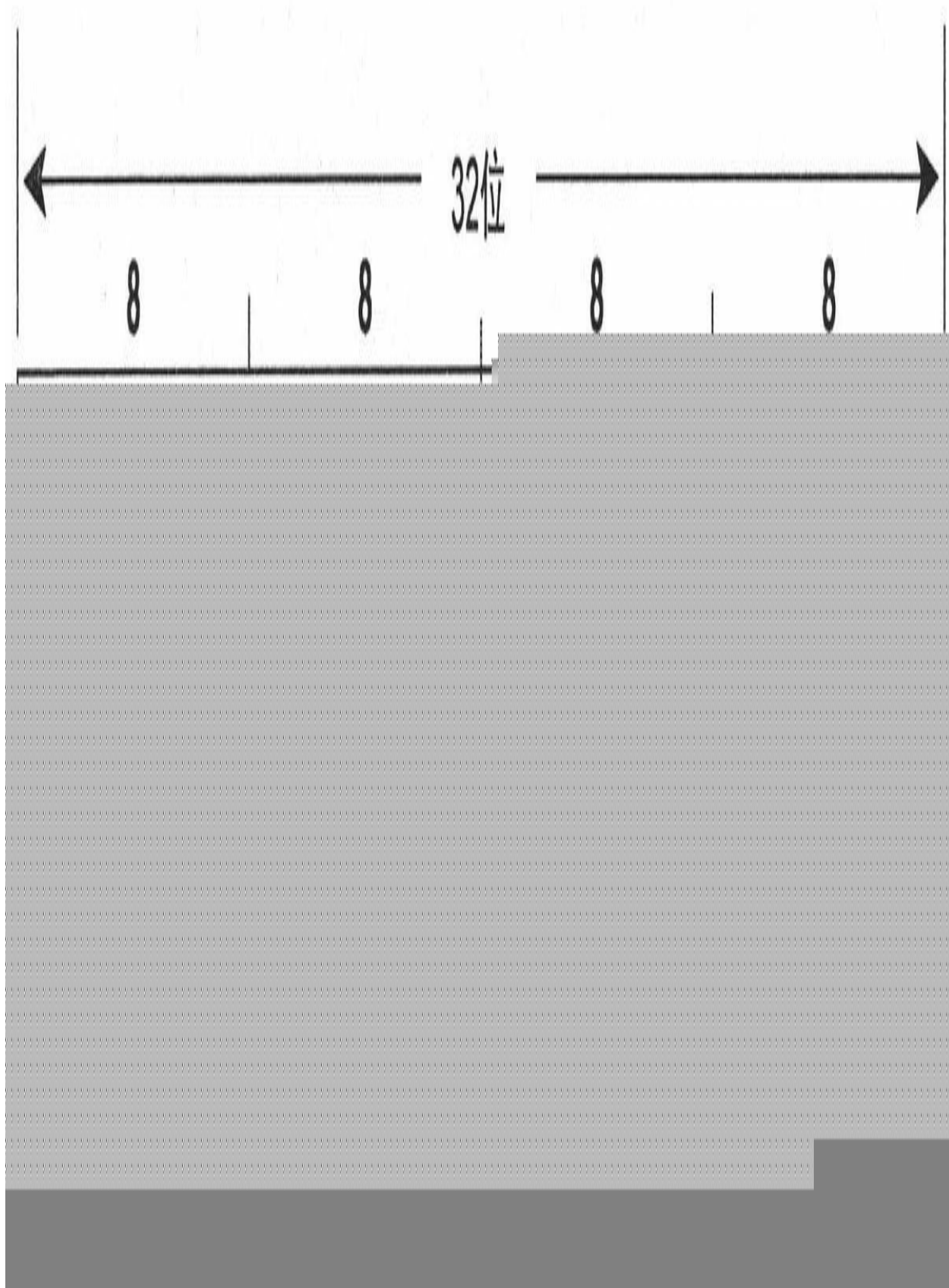
在路由器Barney到Ernest_T的串行链路接口上配置触发扩展特性，具体配置参见示例5-24。

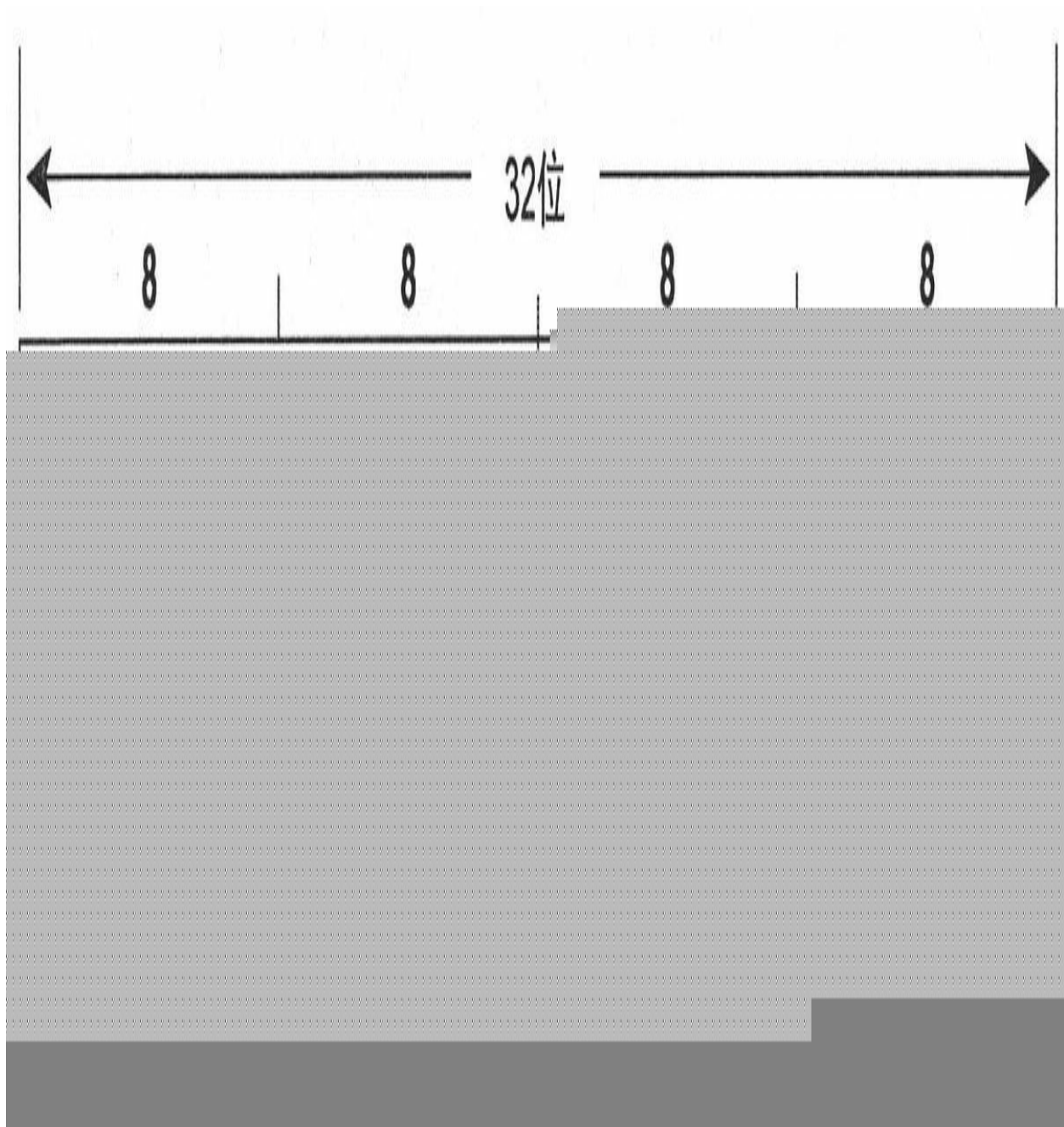
示例5-24 路由器Barney配置了触发更新，而不是周期性的更新



在路由器Barney的调试信息中，显示Barney试图与链路另一端的路由器建立一个触发关系。路由器Barney发送轮询（Poll）并等待确认。当它没有收到任何确认时，路由器Barney开始发送正常的RIPv1更新。调试命令`debug ip rip`和`debug ip rip trigger`的输出参见示例5-25所示。

示例5-25 当一台路由器一开始就配置了触发的RIP后，它会发送轮询信息来确定邻居是否也配置了触发的RIP

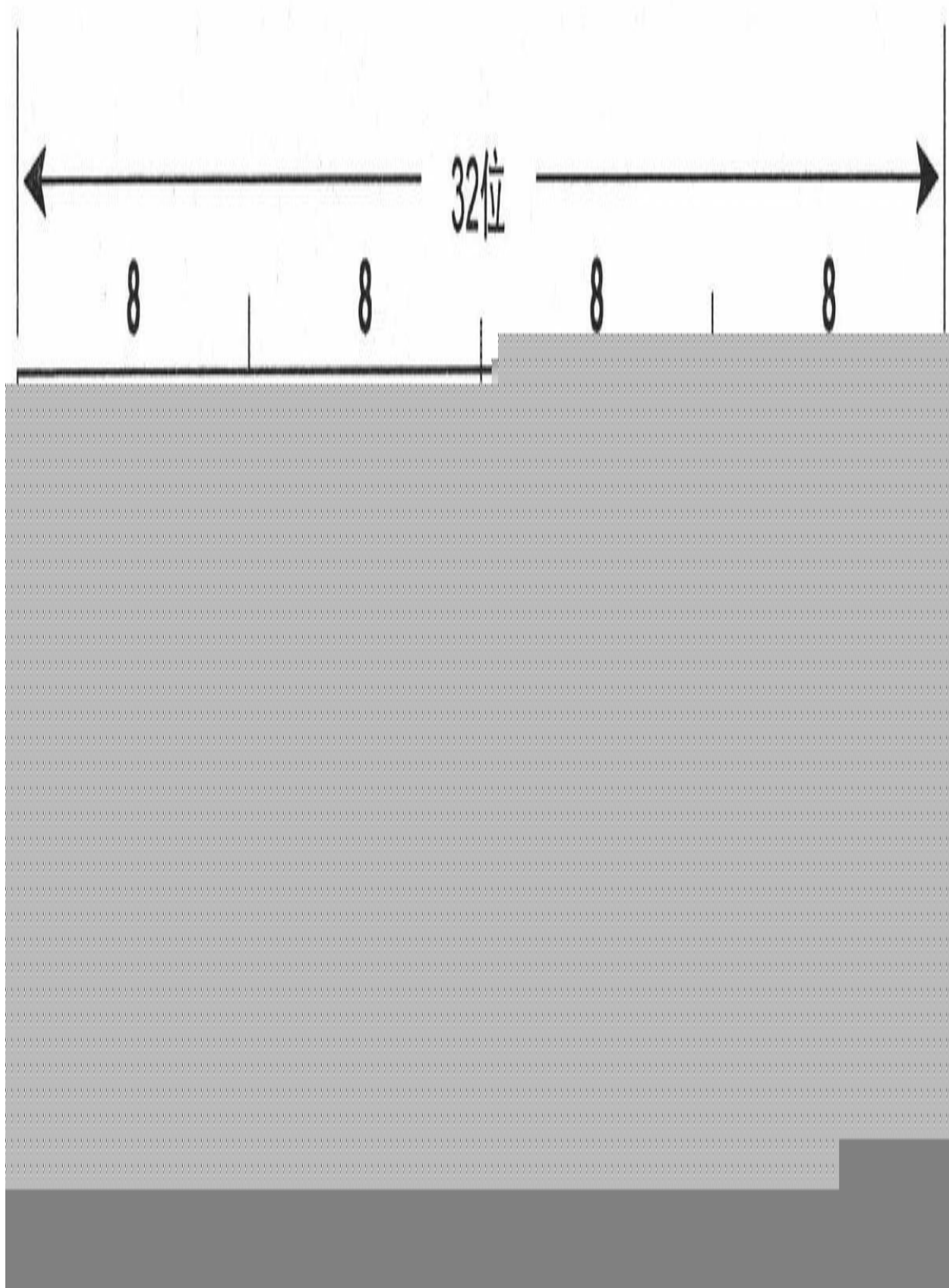




调试输出显示了所配置的路由器发送了6个触发请求。对于每一个请求，路由器都会设置一个轮询计时器，每个周期为5s。如果在5s内还没有收到确认消息，就会发送另一个触发请求。如果在发送完6个触发请求后还没有收到确认消息，那么这个轮询就认为超时了，路由器将等待下一个普通的更新时间，并广播一个RIP更新。在路由器Barney发送触发请求的时候，路由器Ernest_T继续广播它自己的RIP更新。

现在路由器Ernest_T到路由器Barney的串行链路配置了触发的RIP。通过在路由器Ernest_T上的调试，示例5-26中显示了路由器Barney和Ernest_T的初始化过程。

示例5-26 调试输出显示了两台路由器正在建立一个触发的**RIP**关联关系



触发状态从DOWN状态开始，经过INIT和LOADING状态，最后为FULL状态。而后进行路由信息的交换和更新的确认。在输出信息的结尾部分，读者可以看到RIP更新计时器正在超时，但是没有新的更新发送，也没有收到更新。

5.3 RIP故障诊断

RIP协议的故障诊断相对来说是比较简单的。对于RIP这样的有类别路由选择协议来说，最困难的排错就是出现子网掩码配置错误或者子网不连续的情形。如果路由表包含了不准确的或被丢失的路由，那么就应该检查邻近的所有子网和所有子网掩码的一致性。

最后，有一条命令在一台高速路由器向一台低速路由器发送大量RIP消息时可能比较有用。在这种情况下，低速路由器并不能像接收一样快地处理这些路由更新，因此可能会丢失路由信息。这种情况可以通过在RIP的处理中使用Output-delay命令来设置一个8~50ms的发包之间的延迟间隙（缺省为0ms）来解决。

5.4 展 望

RIP协议的简单、成熟和使用的广泛性确保了它还将会使用许多年。但是，读者在本章也可以看到，RIP协议有类别特性的限制。下一章将会继续介绍RIP协议，读者可以了解该协议是怎样扩展来支持无类别路由选择的，也可以了解到为了支持IPv6协议所做的扩展。

5.5 总结表：第5章命令总结

命令	描述
debug ip rip [events]	简要地显示路由器收发的RIP信息
ip address <i>ip-address mask secondary</i>	在接口上指定一个IP地址作为辅助地址
ip rip triggered	在某个接口上配置RIP的触发扩展特性
neighbor <i>ip-address</i>	通过指定接口邻居的IP地址来建立邻接关系
network <i>network-number</i>	指定一个需要运行RIP的网络
offset-list { <i>access-list-number</i> } { <i>in</i> out } <i>offset [type number]</i>	指定路由表中一个与指定的访问列表匹配的路由条目，将自己的度量值增加一个指定的偏移量
output-delay <i>delay</i>	设定一个指定延迟长度的延迟间隙，以便协调高速路由器和低速路由器之间的延迟问题
passive-interface <i>type number</i>	在指定类型和序列号的接口上阻止RIP广播
router rip	启动RIP进程
timers basic <i>update invalid holddown flush</i>	修改指定的计时器的值

5.6 推荐读物

Hedrick, C. “Routing Information Protocol”, RFC 1058, 1988年6月。

Meyer, G. 和 Sherry, S. “Triggered Extensions to RIP to Support Demand Circuits”。

RFC 2091, 1997年1月。

5.7 复习题

1. RIP协议使用什么端口？
2. RIP协议使用什么度量？怎样用度量来表示一个不可达的网络？
3. RIP协议的更新周期是多少？
4. 在一条路由被标记成不可到达之前，必须忽略多少更新？
5. 垃圾收集计时器的用途是什么？
6. 为什么触发更新要使用一个随机计时器？这个计时器的大小范围是什么？
7. RIP协议的请求消息和响应消息之间有哪些不同之处？
8. RIP协议使用哪两种类型的请求消息？
9. 在什么情况下会发出一个RIP协议的响应消息？
10. 为什么RIP协议在主网络的边界处会屏蔽子网？

5.8 配置练习

1. 写出图5-11中所示的6台路由器的相关配置，使它们可以利用RIP协议来为所有的子网进行路由选择。

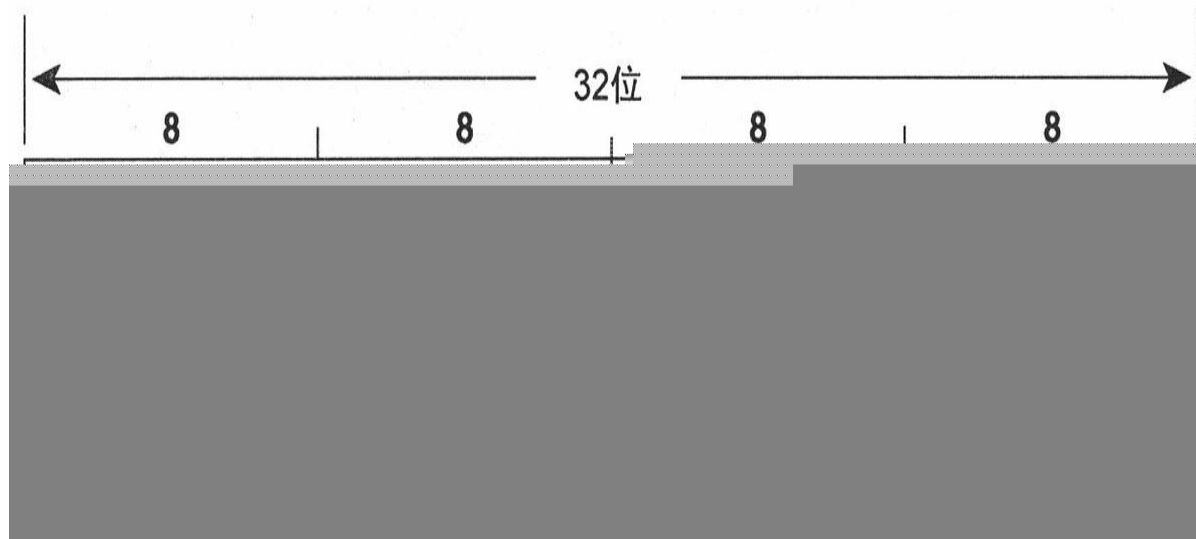


图5-11 配置练习1~4的网络

2. 更改配置练习1中的配置，使路由器RTC和RTD之间RIP更新的通告由广播方式改为单播方式。
3. 在图5-11中，路由器RTC和RTD之间的串行链路的带宽十分有限，请进一步调整RIP协议的配置，使得经过这条链路的RIP更新能够每两分钟发送一次。这里要仔细考虑的是，必须要更改哪些计时器？以及必须要更改哪些路由器上的计时器？
4. 制定一个路由策略，使网络192.168.4.0在路由器RTA看来是不可到达的，而网络192.168.5.0在路由器RTB看来是不可到达的，可以利用偏移列表来实现该策略。
5. 依照“有类别路由选择：直连的子网络”一节所述，在一个主类别分类网络中的所有子网掩码必须是一致的。但那一节中却没有强调一个主类别分类网络内的子网掩码必须是相同的。图5-12中的那两台路由器的RIP配置如下：

router rip

network 192.168.20.0

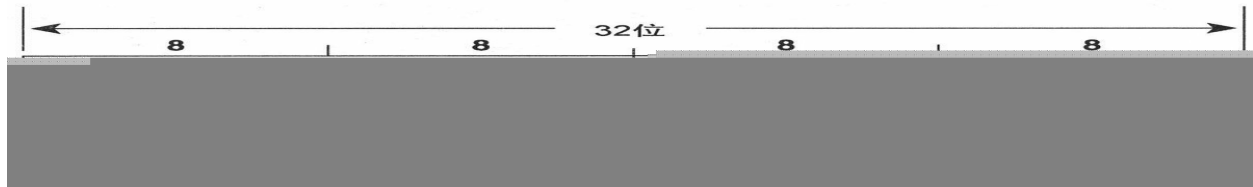
在这个小型的网络中，数据包可以被正确地路由转发吗？解释一下为什么可以或者为什么不可以。



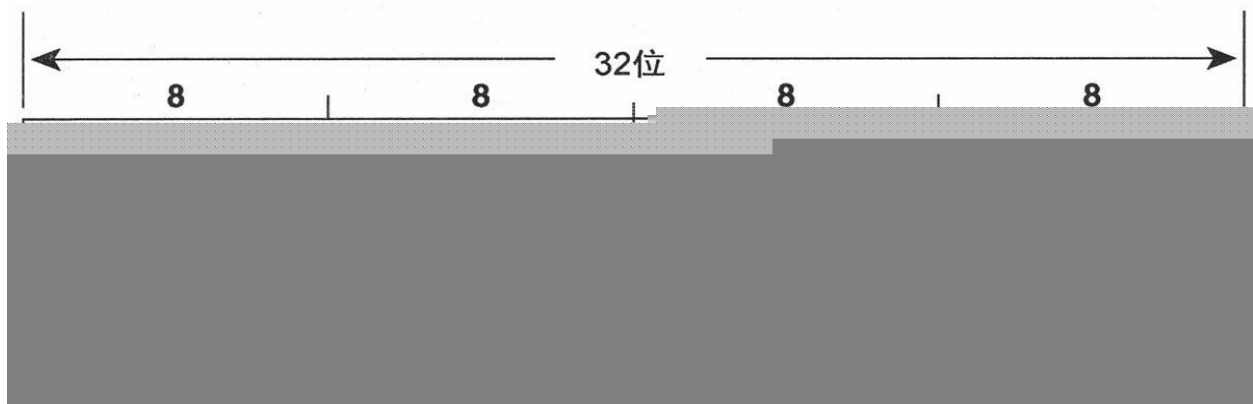
图5-12 配置练习5使用的网络

5.9 故障诊断练习

1. 在第一个偏移列表的例子中，路由器Barney上的访问列表由



更改成



会出现什么结果？

2. 在图5-13中显示了一个网络，其中有一台路由器的IP地址掩码配置错误。在示例5-27～示例5-29中分别显示了路由器RTA、RTB和RTC的路由表。请读者根据前面所了解的关于RIP协议通告和接收路由更新的知识，解释一下路由器RTB的路由表中的每一个路由表项。并请解释一下路由器RTB的路由表中子网172.16.26.0的掩码为什么是32位的？在所有的路由表中，如果存在被丢失的路由条目，请解释为什么？

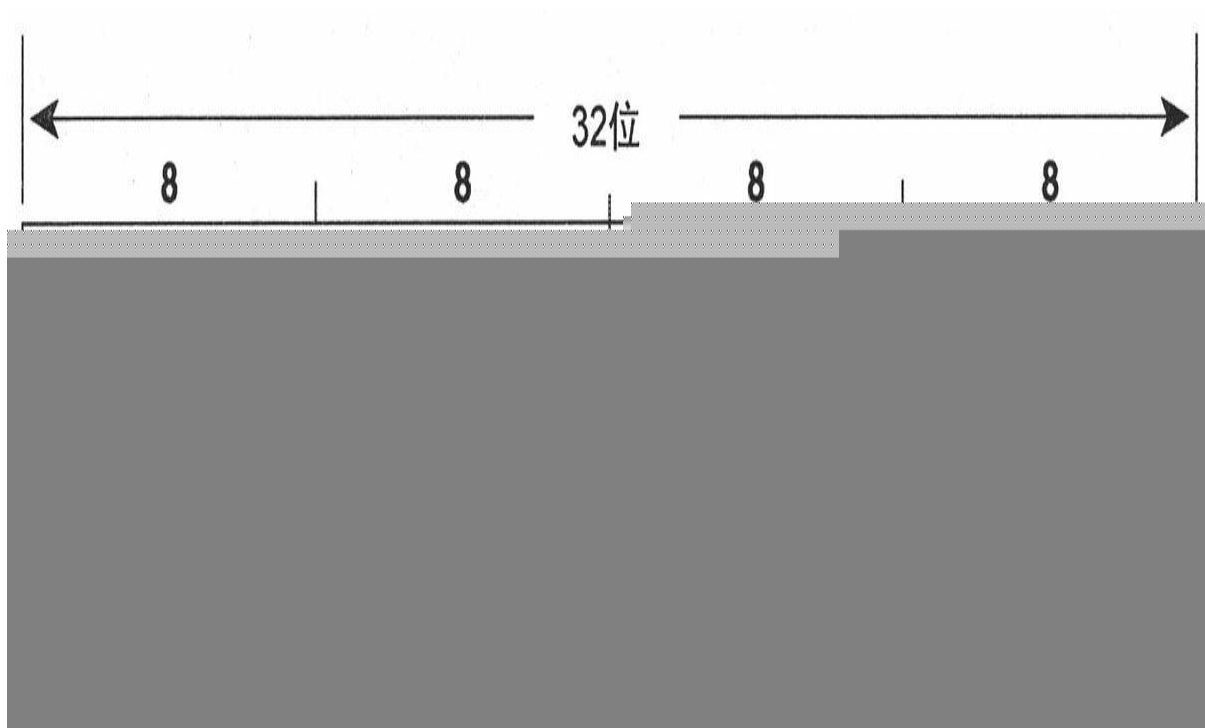
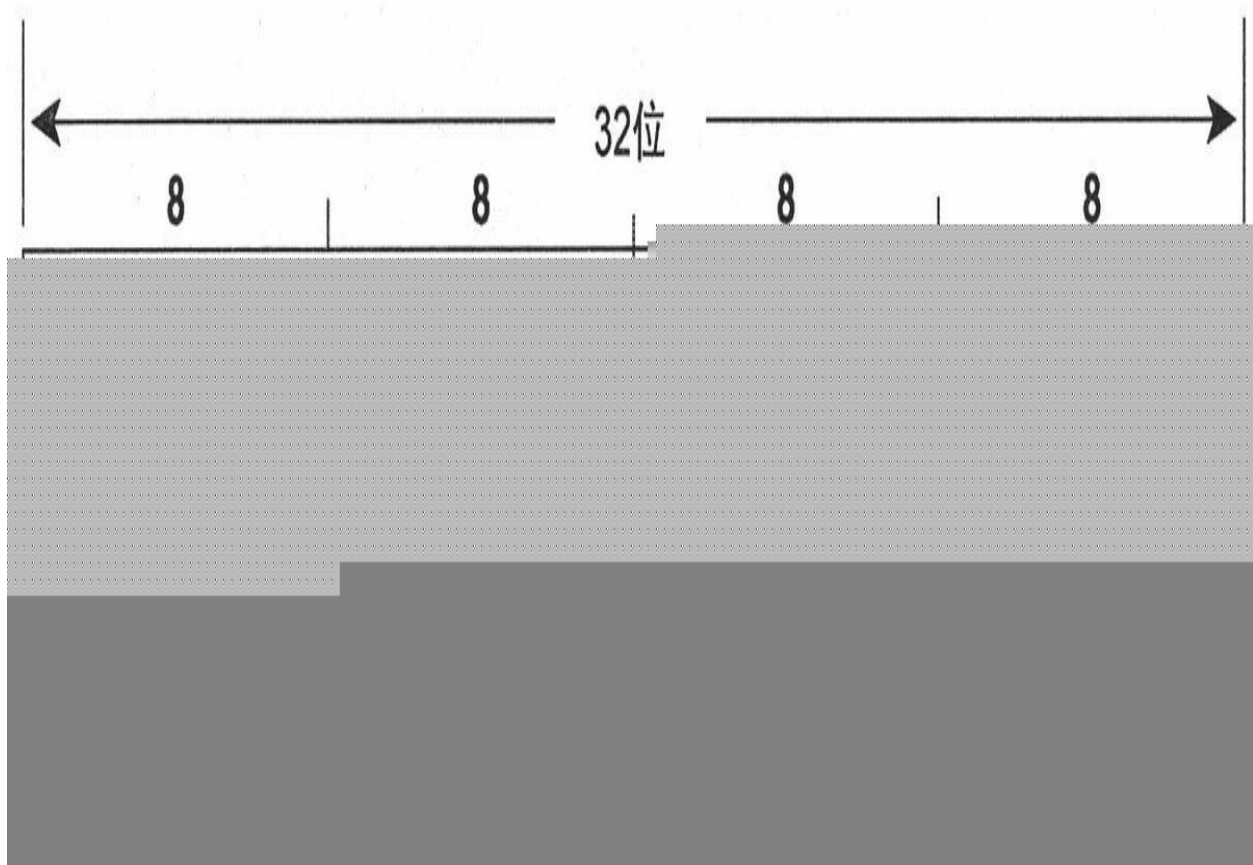
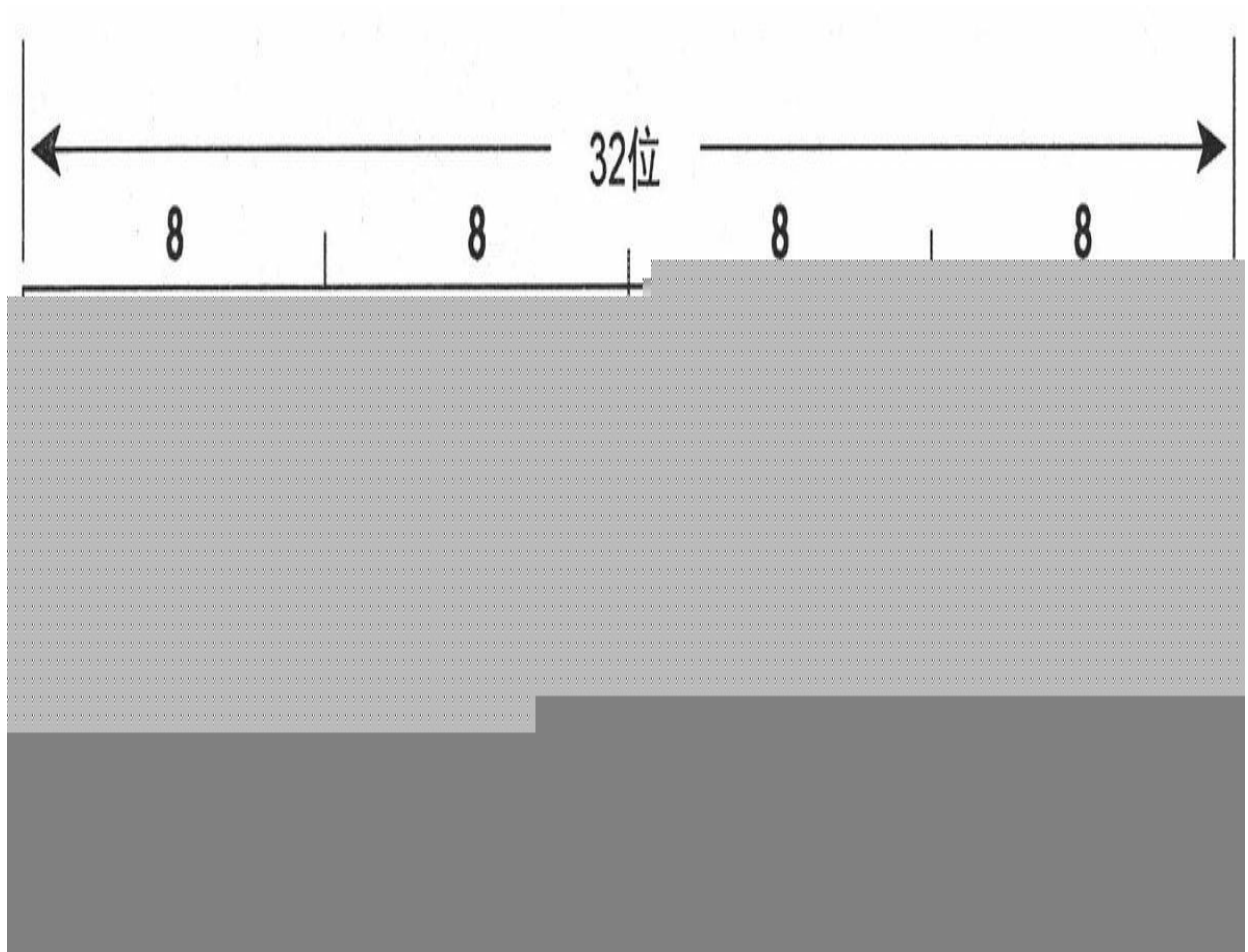


图5-13 故障诊断练习2和3的网络

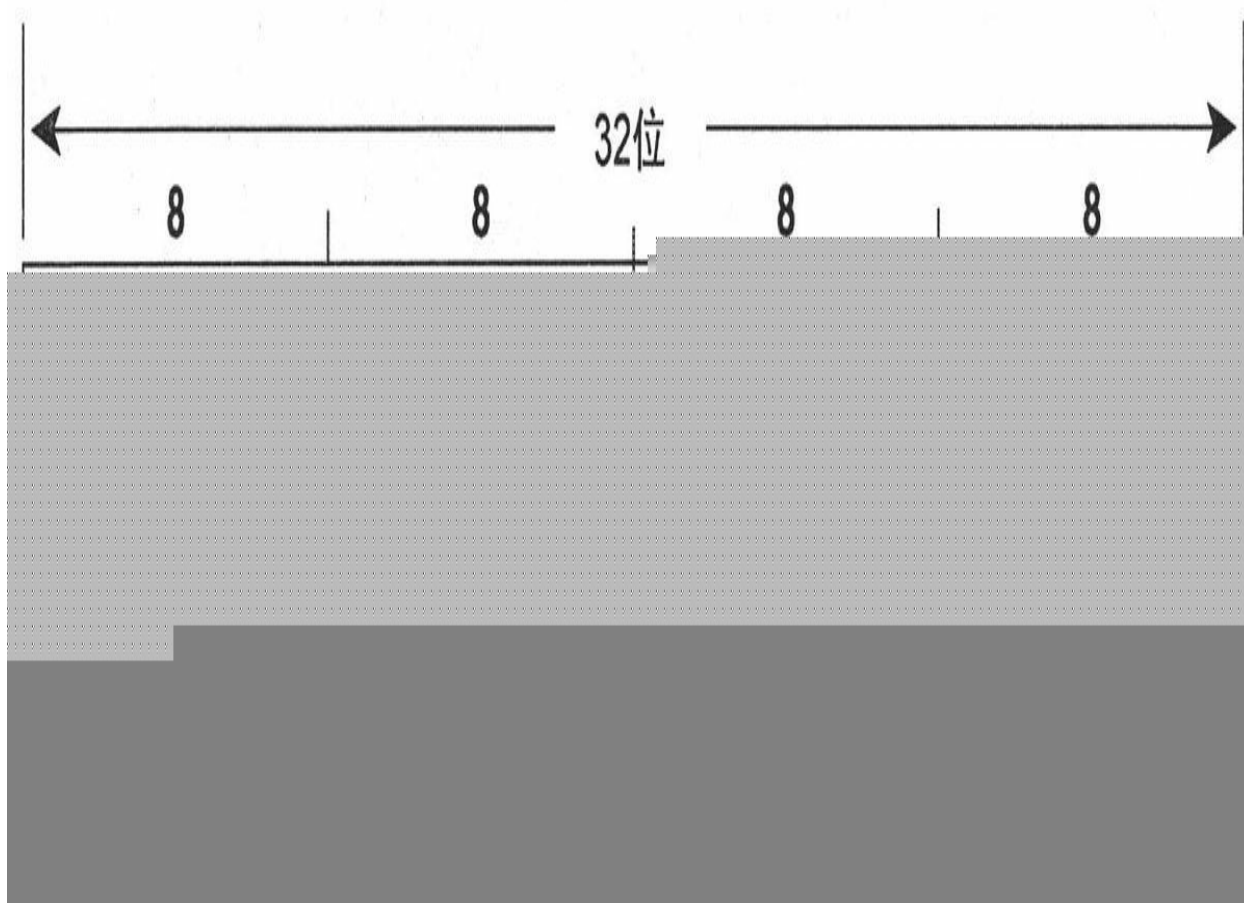
示例5-27 图5-13中路由器RTA的路由表



示例5-28 图5-13中路由器RTB的路由表

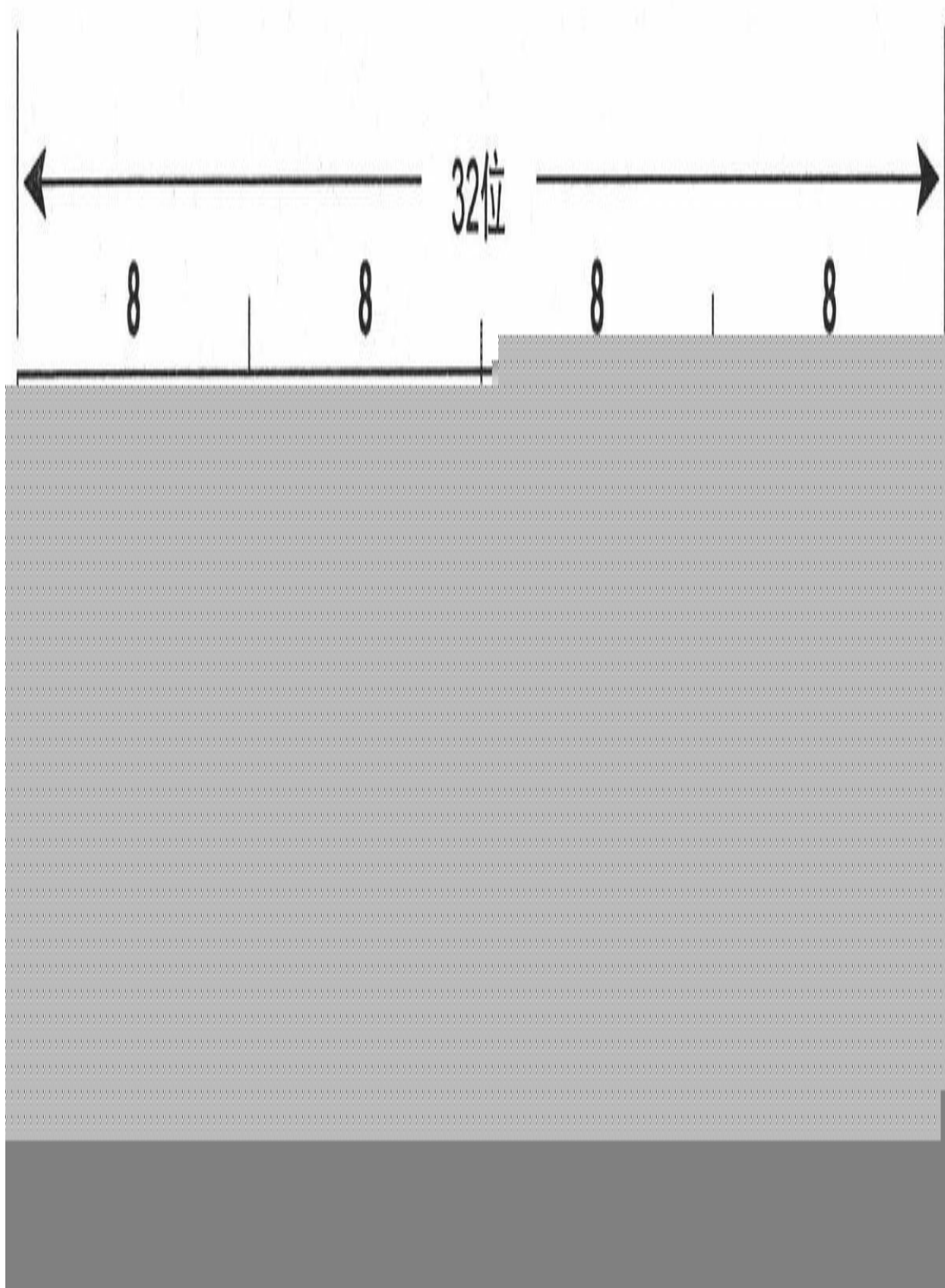


示例5-29 图5-13中路由器RTC的路由表

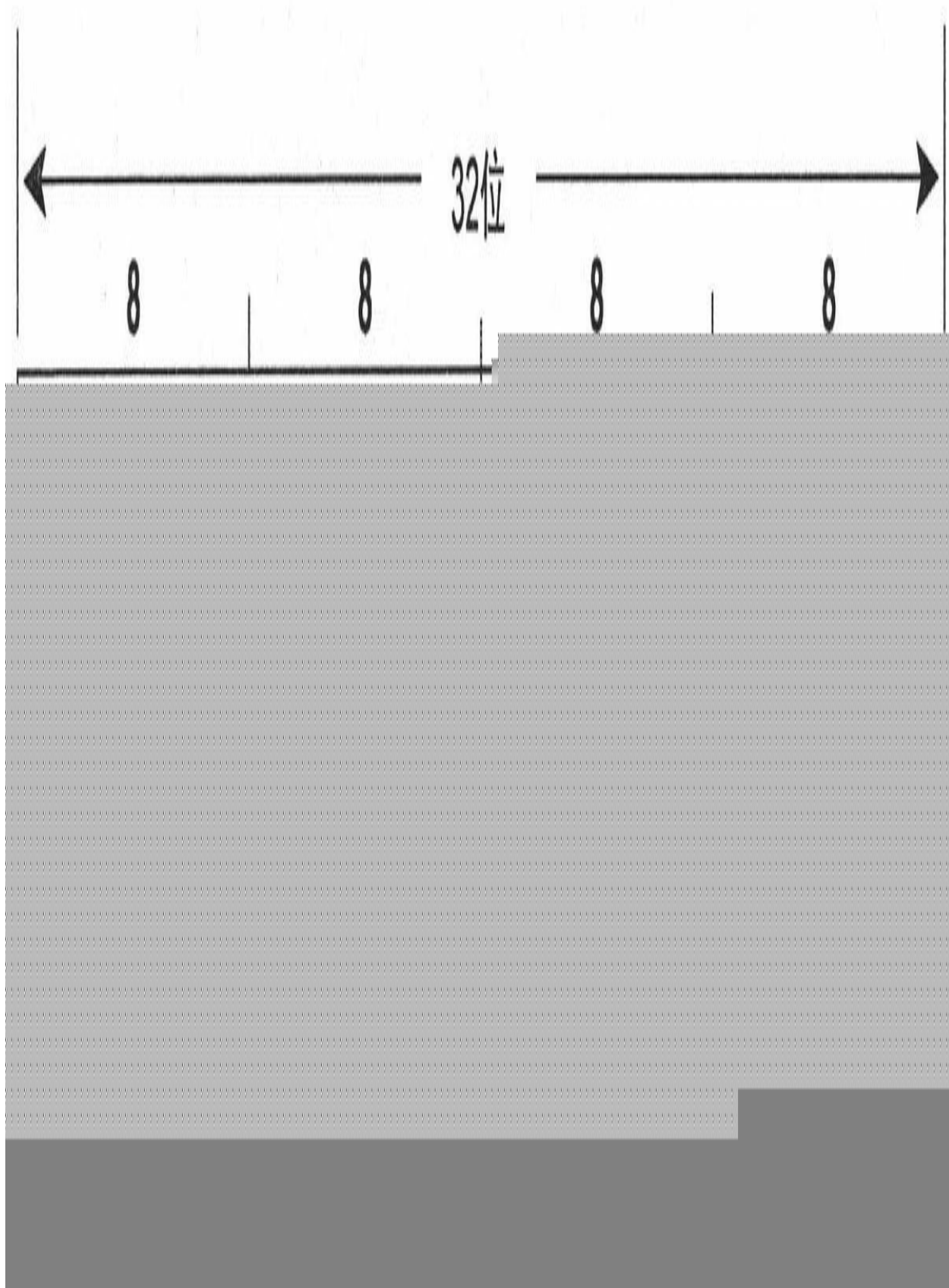


3. 在图5-13中，子网172.16.18.0/23上的用户一直抱怨和子网172.16.26.0/23的连接总是时断时续的——有时通，有时不通（路由器RTB上配置错误的掩码已经改为正确的了）。起初检查路由器RTC和RTD的路由表（参见示例5-30）也没有什么问题，所有的子网都在路由表中。然而经过1min或更长一点的时间，发现路由器RTC显示的子网172.16.26.0/23变得不可到达了（参见示例5-31），而路由器RTD依然显示出所有的子网。再经过几分钟后，路由器RTC的路由表中又看到了那个子网（参见示例5-32）。在显示这3幅图示的每一个的时候，路由器RTD的路由表都没有变化。请仔细检查示例5-30～示例5-32中的路由表所隐含的问题，看看到底是什么问题。

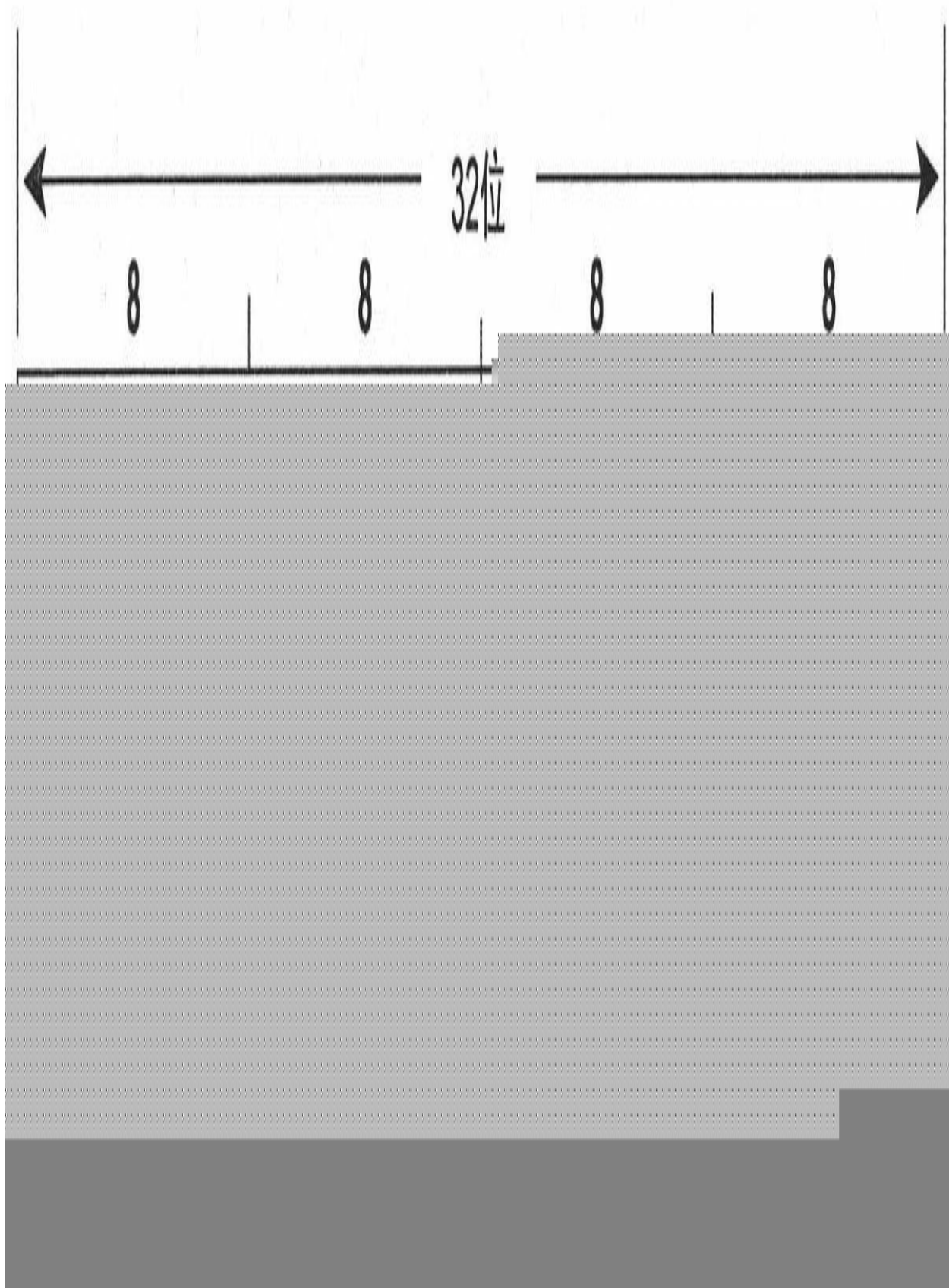
示例5-30 图5-13中路由器RTC和RTD的路由表



示例5-31 在示例**5-30**显示大约**60s**后检查得到的路由器**RTC**和**RTD**的路由表



示例5-32 在示例**5-31**显示大约**120s**后检查得到的路由器**RTC**和**RTD**的路由表



[1] R. E. Bellman. *Dynamic Programming*. Princeton, New Jersey: Princeton University Press; 1957.

[2] L. R. Ford Jr. and D. R. Fulkerson. *Flows in Networks*. Princeton, New Jersey: Princeton University Press; 1962.

[3] Palo Alto Research Center.

[4] 读作“route-dee”和“gate-dee”。

[5] Holddowns用于Cisco IOS，但它不是RFC 1058指定的稳定性特性之一。

[6] 路由表的同步在第4章中讨论。

[7] RIP的一些实现方式也可以只在广播型介质网络上广播，而在点到点的链路上直接发送给对端直连的邻居路由器。Cisco路由器中，如果不改变配置成其他方式的话，RIP更新将在任何类型的链路上广播。

[8] Cisco路由器使用60s的垃圾收集计时器，虽然RFC 1058规定为120s。

[9] neighbor的另外应用是，在像帧中继这样的非广播介质型网络上发送单播更新。

[10] 参照附录B中关于访问列表的讲述。

[11] 触发扩展特性是在RFC 2091中定义的，并在12.0（1）T版本首先引入IOS软件系统。

本章包括以下主题：

- RIPv2的基本原理与实现；
- RIPv2的配置；
- RIPv2和RIPv3的故障诊断。

第6章

RIPv2、RIPng和无类别路由选择

RIPv2协议在RFC 17231中进行了定义，[\[1\]](#)并在Cisco IOS 11.1版及后续的版本中得到支持。确切地说，RIPv2协议不是一个新的协议；它只是在RIPv1协议的基础上增加了一些扩展特性，以适用于现代网络的路由选择环境。这些扩展特性有：

- 每个路由条目都携带自己的子网掩码；
- 路由选择更新具有认证功能；
- 每个路由条目都携带下一跳地址；
- 外部路由标志；
- 组播路由更新。

在这些扩展特性中，最重要的就是路由选择更新条目增加了子网掩码的字段，因而RIPv2协议可以使用可变长的子网掩码，使其成为一个支持无类别路由选择的协议。

RIPv2协议是本书中讲述的第一个无类别路由选择协议。因此，本章将同时介绍无类别路由选择和RIPv2协议。

RIP下一代（RIP next generation, RIPng）是RIPv2对IPv6路由选择的修改，也将在本章中讲述。与IPv4不同，IPv6天生就是无类别的；它没有A类、B类和C类地址，或类似的地址分组。因此，RIPng也是一个无类别路由选择协议。

6.1 RIPv2的基本原理与实现

所有在RIPv1中运用的实现方法、计时器和稳定特性都同样可以在版本2中使用，其中只有一个例外，就是路由更新的广播。RIPv2协议使用组

播的方式向其他宣告RIPv2的路由器发出更新消息，它所使用的组播地址是保留的D类地址224.0.0.9。使用组播方式的好处在于，本地网络上相连的和RIP路由选择无关的设备不再需要花费时间对路由器广播的更新消息进行解析。组播更新将在6.1.2小节进一步阐述。

先来看一下RIP协议的消息格式中提供了哪些版本2的扩展特性，这一节将主要关注RIPv2的操作和这些新增的扩展特性所带来的好处。

6.1.1 RIPv2的消息格式

RIPv2的消息格式如图6-1所示，它的基本结构和RIPv1相同。所有相对于原来协议的扩展特性都是由未使用的字段提供。和版本1一样，RIPv2的更新消息最大可以包含25个路由条目。同样的，与版本1相同的是，RIPv2的操作使用UDP端口520，并且数据报（datagram）的大小（包括一个8字节的UDP头部）最大为512个八位组（octet）。

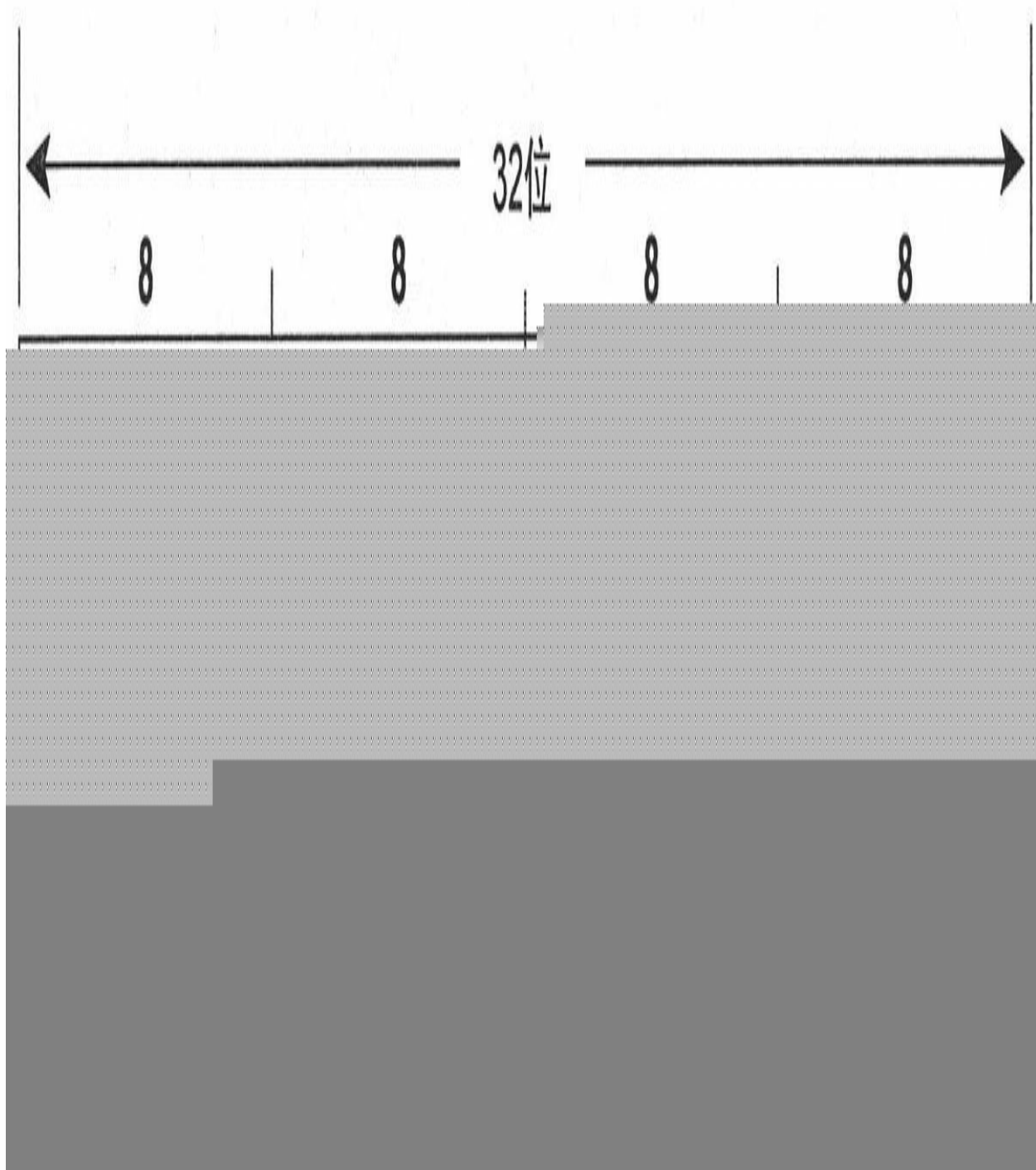


图6-1 RIPv2利用了版本1的消息格式中的未使用字段，因而这些扩展没有改变版本1的基本消息格式

- 命令（**Command**）——取值1和2，1表示为请求消息，2表示响应消息。

- 版本号 (**Version**) ——对于RIPv2, 该字段的值设置为2。如果设置为0或者虽设置为1但消息是无效的RIPv1格式, 那么该消息将被丢弃。RIPv2处理有效的RIPv1消息。
- 地址族标识 (**Address Family Identifier, AFI**) ——对于IPv4协议, 该项总是设置为2。只有一种例外情况, 当该消息对路由器 (或主机) 的整个路由表进行请求时, 这个字段将被设置为0。
- 路由标记 (**Route Tag**) ——提供这个字段用来标记外部路由或重新分配到RIPv2协议中的路由。默认的情况是使用这个16位的字段来携带从外部路由选择协议注入到RIP中的路由的自主系统号。虽然RIP协议自己并不使用这个字段, 但是在多个地点和某个RIP域相连的外部路由, 可能需要使用这个路由标记字段通过RIP域来交换路由信息。这个字段也可以用来把外部路由编成“组”, 以便在RIP域中更容易地控制这些路由。关于路由标记的使用将在第14章中进一步论述。
- IP地址 (**IP Address**) ——路由条目的IPv4目的地址, 它可以是主网络地址、子网地址或主机路由。
- 子网掩码 (**Subnet Mask**) ——是一个32位的掩码, 用来标识IPv4地址的网络和子网部分。这个字段的意义将在6.1.5小节中论述。
- 下一跳 (**Next Hop**) ——如果存在的话, 它标识一个比通告路由器的地址更好的下一跳地址。换句话说, 它指出的下一跳地址, 其度量值比在同一个子网上的通告路由器更靠近目的地。如果这个字段设置为全0 (0.0.0.0), 说明通告路由器的地址是最优的下一跳地址。在本节的最后将用一个例子说明这个字段的用处。
- 度量 (**Metric**) ——是一个在1~16之间的跳数。

图6-2显示了4台连接在同一条以太网链路上的路由器。[\[2\]](#) 路由器Jicarilla、Mescalero和Chiricahua都属于自主系统65501, 并相互宣告RIPv2。路由器Chiricahua是自主系统65501和自主系统65502之间的边界路由器, 在第二个自主系统中, 它把BGP路由通告给路由器Lipan。

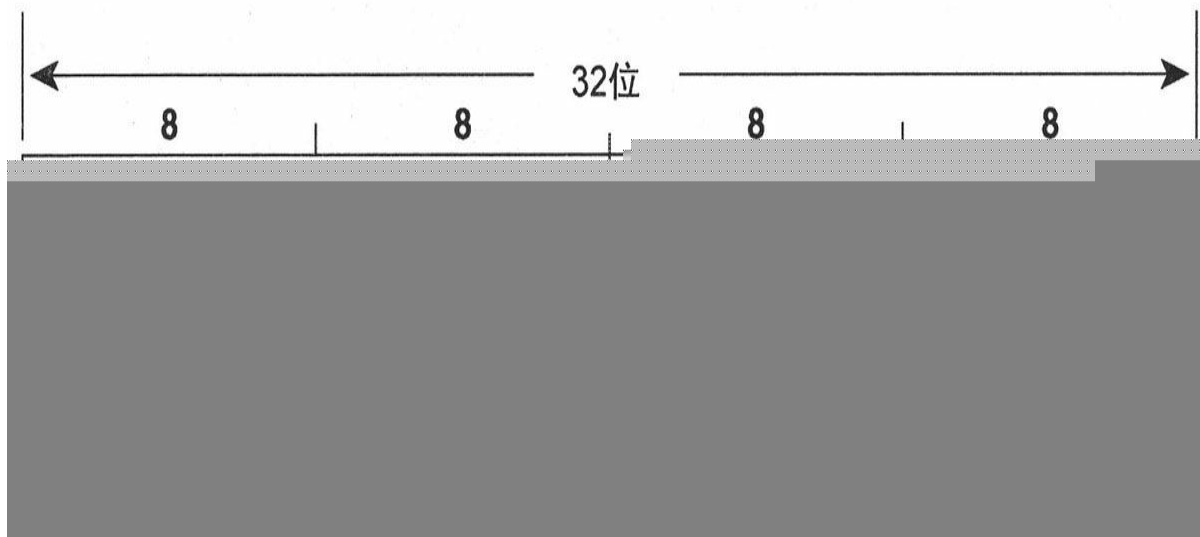


图6-2 虽然它们共享一条公共的数据链路，但路由器Jicarilla和Mescalero只宣告RIPv2，而路由器Lipan只宣告BGP。路由器Chiricahua负责把从后者学习到的路由通告给前面两台路由器

在这里，路由器Chiricahua正在把它从BGP学习到的路由通告给运行RIP的路由器（如图6-3所示）。[3]在路由器Chiricahua的RIPv2通告里，它将使用路由标记字段来指出子网10.3.3.0（掩码是255.255.255.0）是位于自主系统65502（0xFFDE）之中的。路由器Chiricahua也将使用下一跳字段告诉路由器Jicarilla和Mescalero，到达子网10.3.3.0最优的下一跳地址是路由器Lipan的接口10.1.1.3，而不是它自己的接口。注意，由于路由器Lipan不运行RIP协议，而路由器Jicarilla和Mescalero不运行BGP协议，这样即使是在同一个子网中路由器Lipan是可达的，路由器Jicarilla和Mescalero也没有办法直接知道路由器Lipan是最优的下一跳路由器。

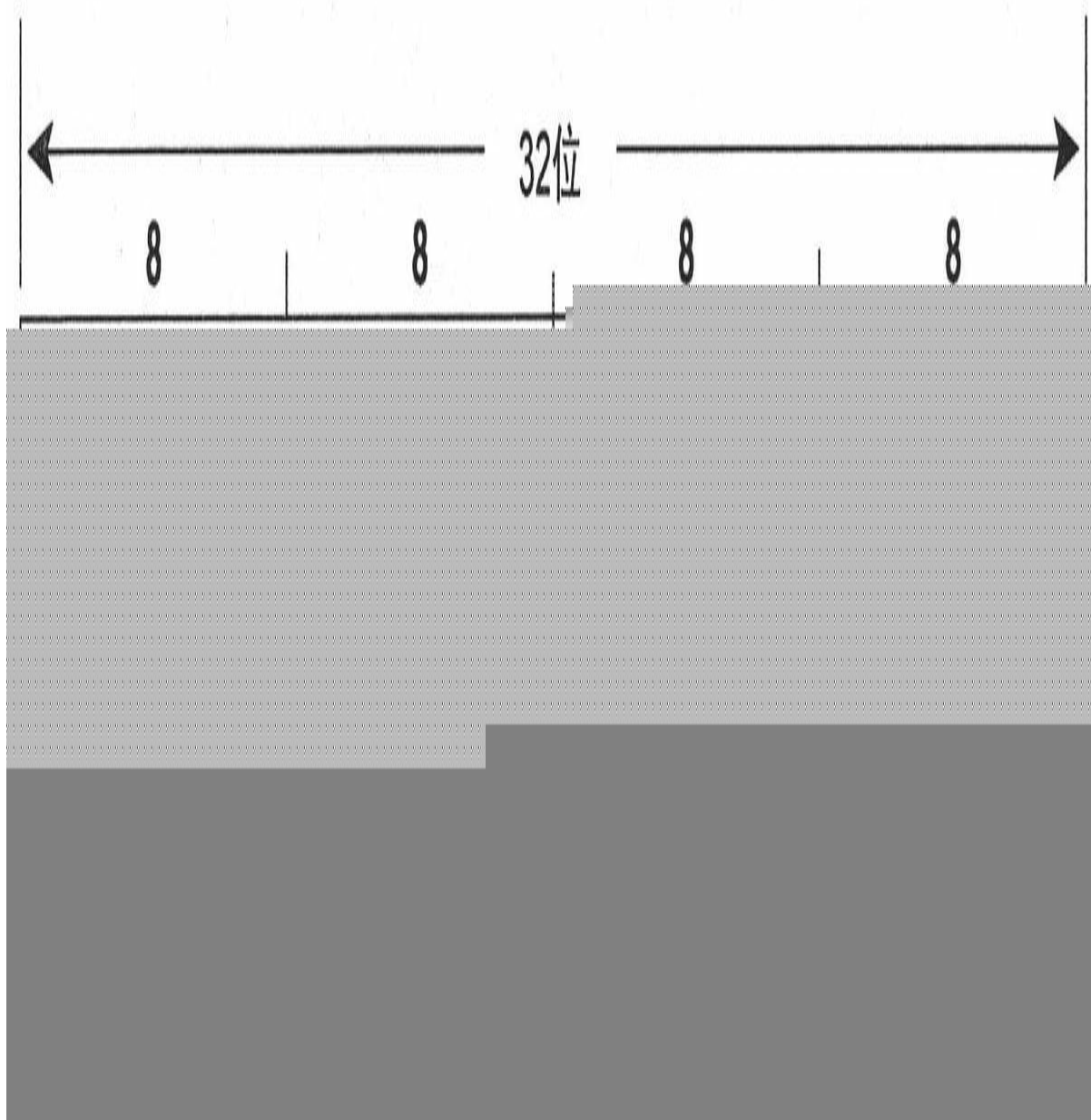


图6-3 从路由器Chiricahua上捕获的RIPv2更新信息显示了通告子网10.3.3.0时所使用的路由标记、子网掩码和下一跳字段

[6.1.2 与RIPv1的兼容性](#)

RIPv1使用灵活的方式来进行路由更新。如果更新消息的版本字段指出RIP是版本1，但所有未使用字段（unused field）的所有位都被设置为1，那么这个更新消息将被丢弃；如果版本字段设置大于1，在版本1中定义为未使用的字段将被忽略，但会处理这个消息。结果是，像RIPv2

这种新版本协议就可以向后兼容RIPv1。

RFC1723用4个设置定义了一个“兼容性开关”，用来允许版本1和版本2之间的互操作：

- **RIP-1** ——只有RIPv1的消息被传送。
- **RIP-1兼容性** ——RIPv2使用广播方式代替多播方式来通告消息，以便RIPv1可以接收它们。
- **RIP-2** ——RIPv2使用多播方式将消息通告到目的地址224.0.0.9。
- **None** ——不发送更新。

RFC建议这些开关基于接口模式上进行配置。在6.3节中指出采用Cisco的命令可以将它们的值设置为1~3，将值设置为4已由**passive-interface**命令完成。

另外，RFC1723定义了一个“接收控制开关”来控制更新的接收。对于这个开关，4种建议的设置是：

- **RIP-1 Only;**
- **RIP-2 Only;**
- **Both;**
- **None。**

此开关也是基于每一个接口上配置的。在6.3节中指出了采用Cisco的命令可以将它们的值设置为1~3，将值设置为4可以通过使用访问列表来过滤UDP源端口号520，或者在该接口上不包含**network** 语句，[\[4\]](#)或者配置一个路由过滤列表来完成，后者将在第13章中讨论。

[6.1.3 无类别路由查找](#)

第5章阐述了有类别路由的查找方法——首先将目的地址与路由选择表

中的主网地址匹配，然后匹配主网的子网。如果经过这些步骤找不到匹配项，这个数据包就会被丢弃。

在IOS11.3版本之前，IOS的缺省方式都是有类别路由查找，但缺省路由的查找改成了无类别查找。对于早期的IOS版本，即使像RIPv1和IGRP这样的有类别路由选择协议，读者也可以通过全局命令**ip classless** 来启用无类别路由查找。当路由器执行无类别路由查找时，它不会注意目的地址的类别，替代方式是，它会在目的地址和所有已知的路由之间逐位（bit-by-bit）执行最佳匹配。当和缺省路由一起工作时，这个特性将变得非常有用，这将在第12章中讨论。连同其他一些无类别路由选择协议的特性，无类别路由查找的功能将显得更加强大。

6.1.4 无类别路由选择协议

无类别路由选择协议最根本的特点是，它可以在路由通告中具有携带子网掩码的能力。每条路由拥有子网掩码的一个好处就是，全0和全1的子网现在可以利用了。在第1章中说明了有类别路由选择协议不能区分全0子网（例如172.16.0.0）和主网络号（172.16.0.0）；同样地，它们也不能区分全1子网的广播（172.16.255.255）和全部子网的广播（172.16.255.255）。

如果包含了子网掩码，这个困难就不存在了。我们可以容易地看出网络172.16.0.0/16是一个主网络号，而172.16.0.0/24则是一个全0的子网。172.16.255.255/16和172.16.255.255/24也可以同样地区分开来。

在缺省的条件下，即使运行的是无类别路由选择协议，Cisco IOS软件也将拒绝试图把一个全0的子网配置为有效的地址/掩码组合。为了使这个缺省的行为无效，可以使用全局命令**ip subnet-zero** 进行更改。

每条路由拥有子网掩码的另一个很大的好处就是，可以使用可变长子网掩码和利用一条单一的聚合地址来汇总一组主网络地址。可变长子网掩码将在下一小节讲述，地址聚合（或超网）将在第7章中介绍。

6.1.5 可变长子网掩码（VLSM）

如果每一个目的地址都可以单独地携带相关联的子网掩码通告到整个网络中，那么就没有什么理由要求所有的掩码必须是等长的了。这个事实

就是可变长子网掩码（VLSM）的基础。

图6-4显示了一个可变长子网掩码的简单应用。图中网络的每一条数据链路都必须有一个惟一标识的子网掩码，同时，每一个子网地址段必须包含足够的主机地址数来提供给这条数据链路上相连的设备使用。

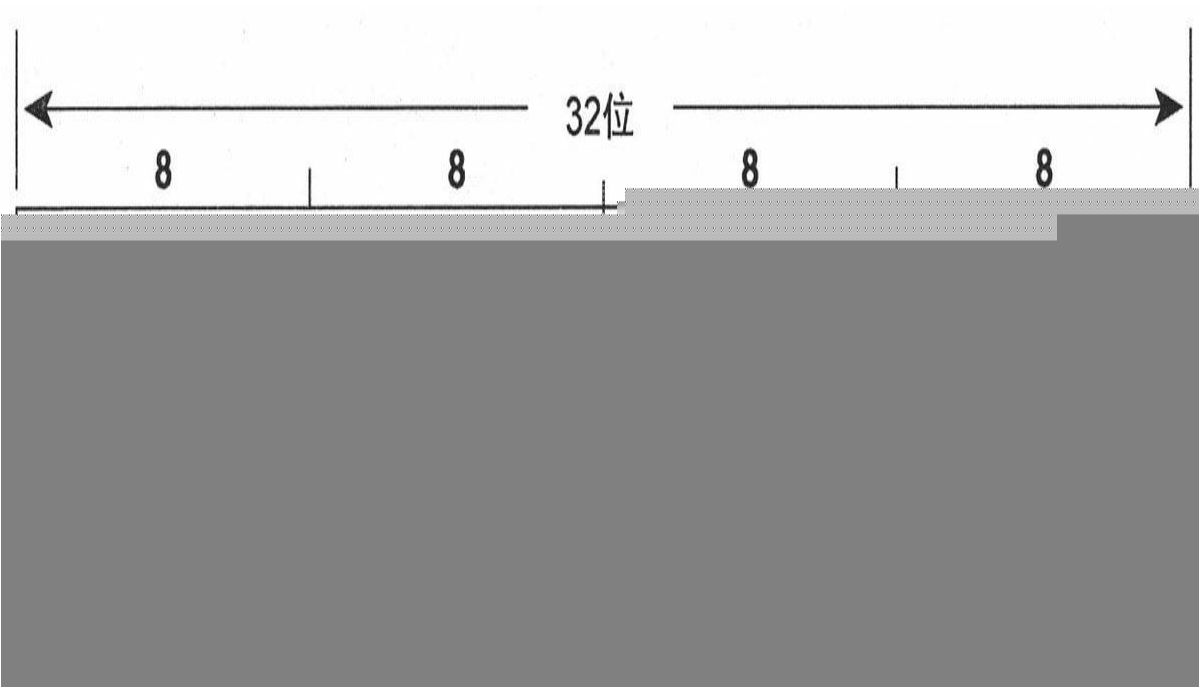


图6-4 使用VLSM，C类地址可以被分成子网，以便提供给这个网络和它的每一条数据链路上的主机使用

考虑分配给这个网络的C类网络地址，没有VLSM技术是不能完成子网划分的。这里令牌环的那个子网需要100个主机地址，也就是需要一个25位的掩码（有1位被子网化）；任何更长一点的掩码都将无法保留有足够的主机位。但是如果所有的子网掩码都必须是等长的话，那么从这个C类地址中只能再分出一个子网。[\[5\]](#)这样将没有办法提供足够的子网数以满足需要。

使用VLSM，在图6-4的网络中不相的主机地址只需要1个C类的网络地址就可以满足了。表6-1显示了这些子网和每个子网内可用的地址范围。

表6-1 图6-4的子网

子网/掩码	地址范围	广播地址
192.168.50.0/25	192.168.50.1-192.168.50.126	192.168.50.127
192.168.50.128/26	192.168.50.129-192.168.50.190	192.168.50.191
192.168.50.192/27	192.168.50.193-192.168.50.222	192.168.50.223
192.168.50.224/28	192.168.50.225-192.168.50.238	192.168.50.239
192.168.50.240/30	192.168.50.241-192.168.50.242	192.168.50.243
192.168.50.244/30	192.168.50.245-192.168.50.246	192.168.50.247

很多人，包括许多使用VLSM的人利用这项技术解决的问题比上述例子复杂得多。使用VLSM技术的关键之处就是，当一个网络地址依据某种标准方式被划分成子网以后，那些子网本身还能够进一步被子网化。事实上，有时候我们会偶尔听到VLSM关于“子网的子网化”的说法。

仔细察看表6-1的地址（通常是二进制的），就可以发现VLSM是怎样工作的。[\[6\]](#)首先，使用一个25位的掩码把网络地址划分成两个子网：192.168.50.0/25和192.168.50.128/25。第一个子网可以提供126个主机地址满足图6-4中令牌环网段的需要。

根据第1章所讲述的，我们了解到划分子网可以包括扩展的缺省网络掩码，因此一些主机位可以被视为网络位。这种做法同样可以应用于剩下的子网192.168.50.128/25。其中一个以太网段需要50个主机地址，因此余下的子网掩码应该扩展到26位，这一步提供了两个子网的子网（sub-subnets）——192.168.50.128/26和192.168.50.192/26，每一个小子网都可以提供62个可用的主机地址。第一个小子网可以给前面那个大的以太网段使用，剩下的第二个小子网可以进一步子网化提供给其他数据链路使用。

这个过程再重复两次，就可以提供必需的满足大小的子网给小的以太网段和FDDI环网段使用。剩余的子网192.168.50.240/28可以用作两个串行链路所需要的子网。对于任何点到点的链路，最普遍的情况只需要两个主机地址——每端一个地址。使用30位的掩码可以创建这两个串行链路的子网，每个子网正好包括两个可用的主机地址。

点到点的链路需要子网地址，但每个子网只需要两个主机地址，这就是使用VLSM的一个理由。例如，图6-5显示了一个典型的广域网络的拓扑结构，它将每一个远程路由器都通过帧中继PVC与中心路由器（hub

router) 相连。现代网络设计的经验通常建议在点到点的子接口^[7]上配置每个PVC电路。如果没有VLSM技术，将必须有相同大小的子网，而这个子网的大小却是主机设备数量最多的子网所需要的地址数。

假设图6-5中的网络使用了一个B类地址，并且每一台路由器都和几个局域网相连，而这些局域网连接的设备数量最大为175台。包含每个PVC电路的子网都需要一个24位的掩码，这样对网络中的每一个PVC链路来说，就有252个地址浪费了。使用VLSM技术，选用单个子网并使用30位的掩码对子网进行子网化就可以满足需要了，可以划分足够的子网最多可以满足64个点到点的链路（如图6-6所示）。

VLSM地址设计的例子在本章和后续的章节中都有描述。第7章介绍了使用VLSM技术的另一个主要用途，即层次化的编址和地址聚合。

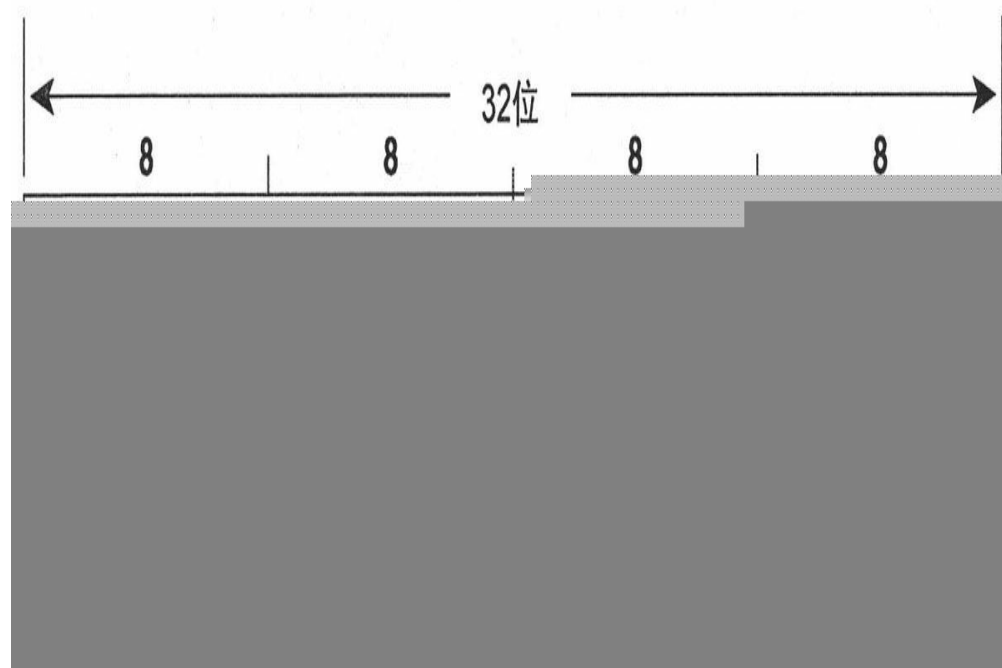


图6-5 VLSM允许这些PVC虚电路中的每一个电路都配置一个单独的子网，而不浪费主机地址

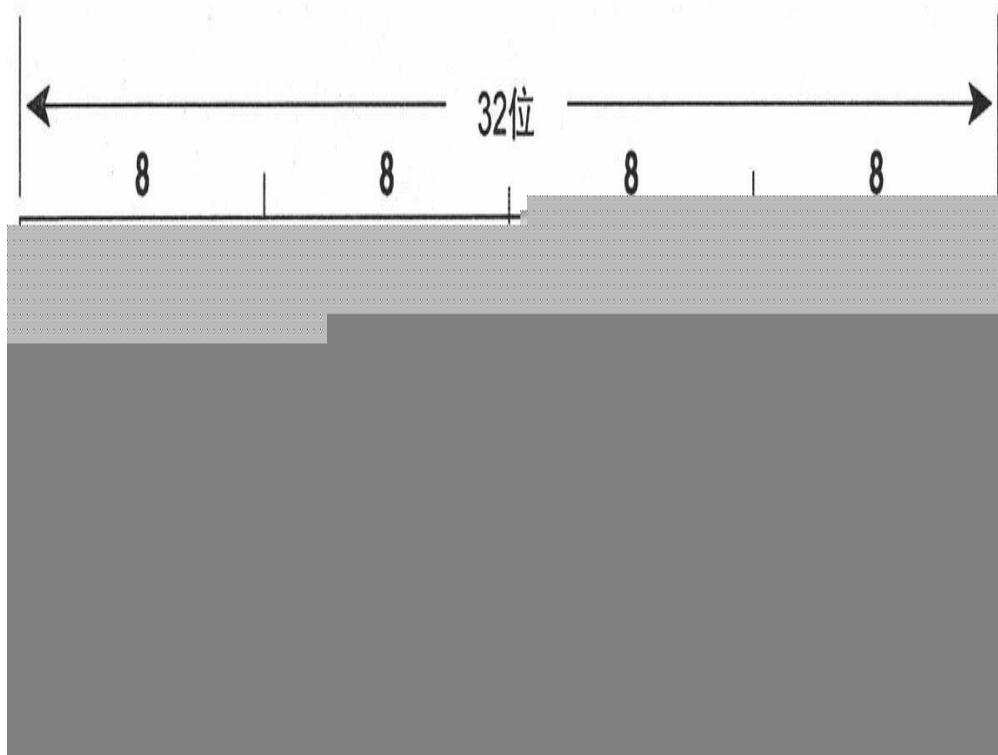


图6-6 这个B类地址使用一个24位掩码来划分子网，对子网172.17.11.0使用30位的掩码进一步地划分更小的子网，结果可以为相关的点到点链路提供64个子网

6.1.6 认证

涉及到所有路由选择协议的安全问题，是指一台路由器接受非法路由选择更新消息的可能性。非法的更新消息可能来自试图破坏网络的攻击者，或试图通过欺骗路由器发送数据包到一个错误的目的地的方法来捕获数据包。更普遍的有害更新消息来自出现故障的路由器。RIPv2协议能够通过更新消息所包含的口令来验证某个路由选择更新消息源的合法性。

RIPv2是通过更改RIP消息中正常情况下应该是第一个路由条目的字段来支持认证的，如图6-7所示。在含有认证的单个更新消息中，最大可以携带的路由条目被减少到了24个。认证是通过设置地址族标识字段为全1（0xFFFF）来标识的。对于简单的口令认证，认证的类型

（authentication type）是2（0x0002），剩余的16个八位组字节字段携带一个最多有16个字符的口令，可以由数字和字母混合组成。口令在字段

中按照左对齐的方式，如果一个口令小于16个八位组字节的长度，那么字段中没有使用的位被设置成0来填充。

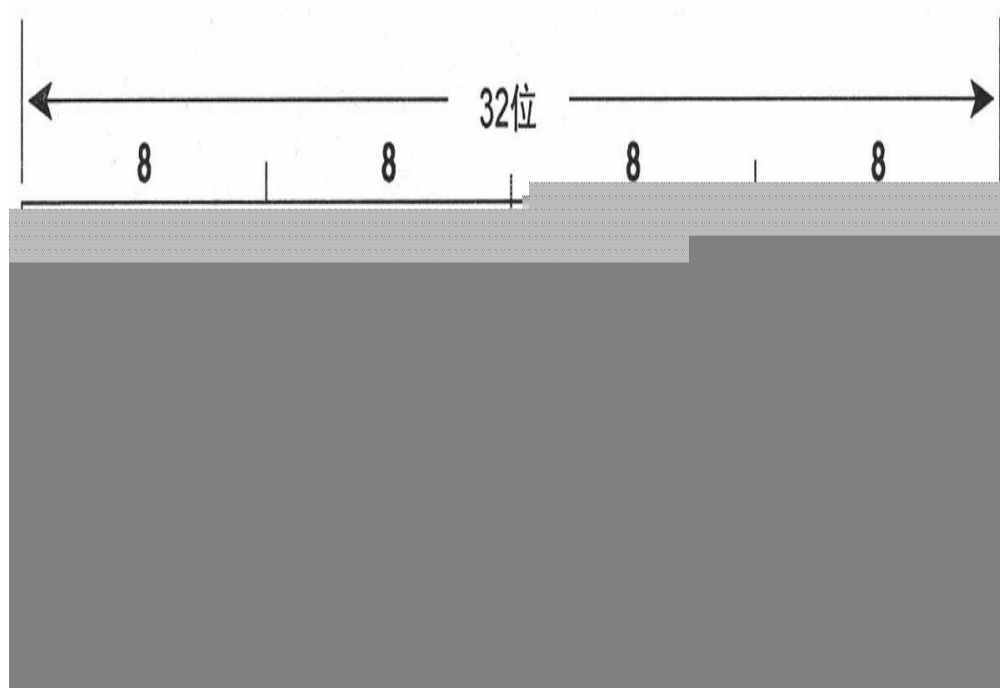


图6-7 RIPv2的认证信息，配置是加载在第一个路由条目的字段空间上的

图6-8显示了协议分析仪捕获到的一个包含认证的RIPv2消息。该图也显示了使用缺省的RIP认证的一个安全隐患：口令是以明文方式传输的。任何人捕获到包含RIPv2更新消息的数据包，都可以读出它的认证口令。

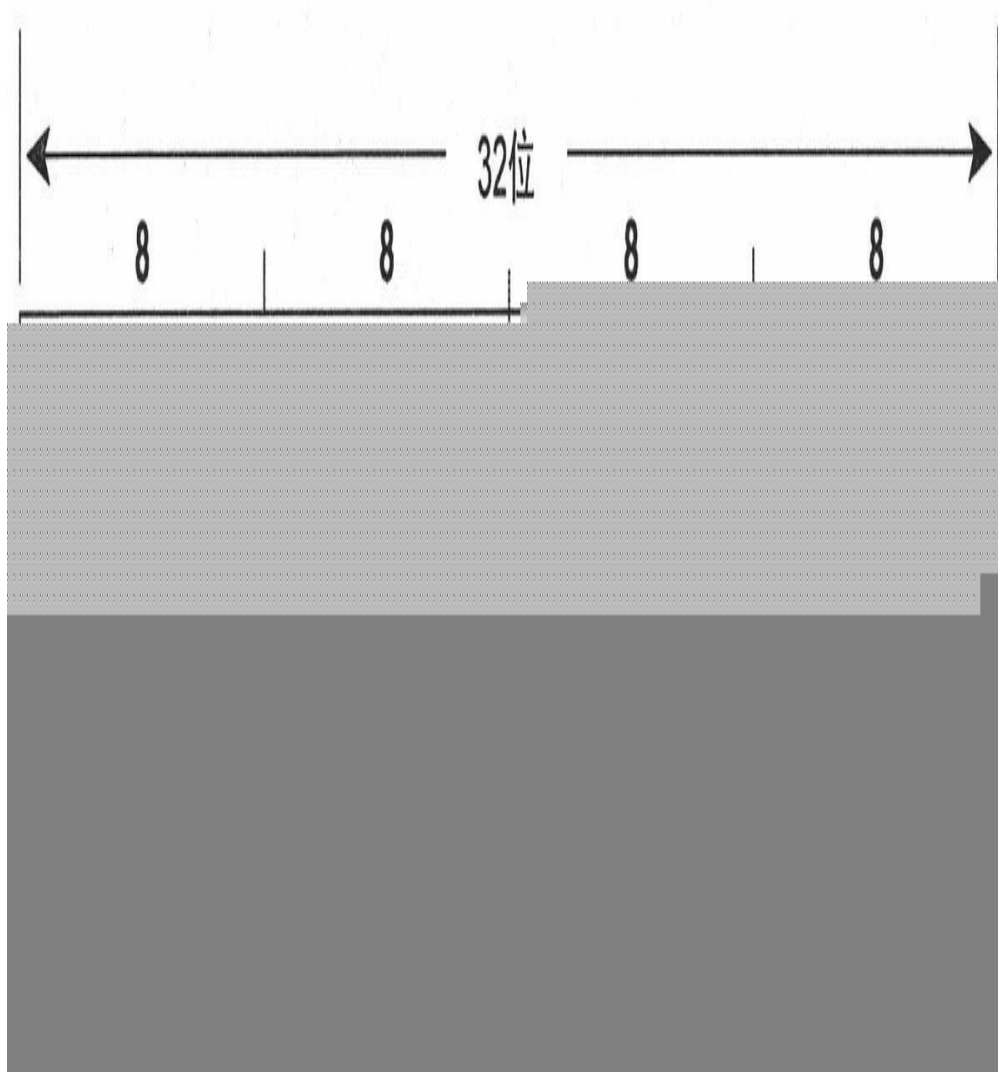


图6-8 当使用简单的口令认证时，口令是以明文方式传输的，因而可以被任何人通过侦探包含更新的数据包读出口令

虽然RFC1723只描述了简单口令认证，但却很有远见地提供了一个认证类型字段。Cisco IOS软件就是利用这个特定字段，提供了用MD5认证来替代简单口令认证的选项。[\[8\]](#)

Cisco使用了第一个和最后一个路由条目的字段空间，从而达到了MD5认证的目的。

MD5是一个单向的消息摘要（message digest）算法或安全散列函数（secure hash function），由RSA Data Security, Inc提出。有时候MD5也被作为一个加密校验和（cryptographic checksum），因为它的工作方

式和算术校验和（arithmetic checksum）有点相似。MD5算法是通过一个随意长度的明文消息（例如，一个RIPv2的更新消息）和口令计算出一个128位的hash值的。这个“指纹”随同消息一起传送。拥有相同口令的接收者会计算它自己的hash值，如果消息的内容没有被更改，接收者的hash值应该和消息中发送者的hash值相匹配。

图6-9显示了图6-8中的同一台路由器的更新消息，但是含有MD5认证。这里的认证类型是3，并且看不到口令。

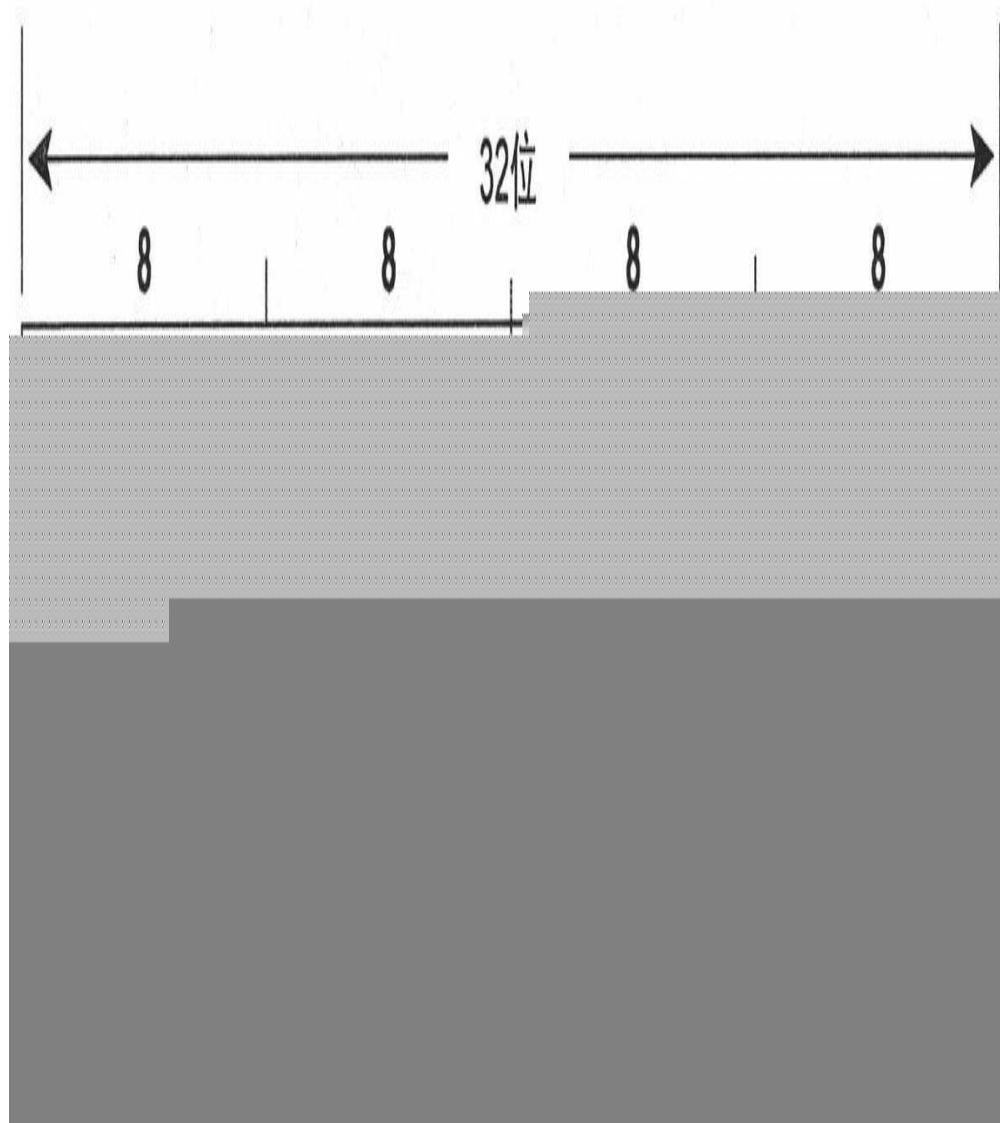


图6-9 这个更新消息和图6-8中的更新来自于同一台路由器，但是它使用了MD5认证

6.2 RIPng的基本原理与实现

支持IPv6的RIPng协议虽然是基于RIPv2协议的，但它并不是RIPv2的简单扩展，它实际上是一个完全独立的协议。RIPng协议不支持IPv4，因此读者如果同时在IPv4和IPv6环境里使用RIP作为路由选择协议，就必须运行支持IPv4的RIPv1或RIPv2，以及支持IPv6的RIPng。

RIPng使用与RIPv2相同的计时器、过程处理和消息类型。例如，RIPng像RIPv2一样，使用30s的更新计时器抖动来避免消息同步，还有180s的超时周期、120s的垃圾收集计时器和180s的抑制计时器。它也使用相同的跳数度量，16跳表示不可到达。RIPng也用与RIPv2相同的方式使用请求和响应消息。另外，除了类似于RIPv1和RIPv2一样用到少数单播方式外，像RIPv2一样，RIPng大多是以多播方式收发请求和响应消息。RIPng使用的IPv6多播地址是FF02::9。

除了上述这些类似的功能外，一个例外之处是认证功能。RIPng本身并没有认证机制，但是承担认证功能的特性已经集成到IPv6中了。

当然，RIPng也不需要像RIPv2那样要求具有对RIPv1的兼容性开关，因为它本来就不向后支持IPv4协议。

如图6-10所示，图中显示了RIPng的消息格式。和RIPv1与RIPv2运行在UDP端520上不同，RIPng发送和接收消息都是运行在UDP端口521上的。与RIPv1和RIPv2的另一个不同之处是它没有设定消息的大小。在这里，消息的大小仅仅依赖于发送它的链路的MTU值。

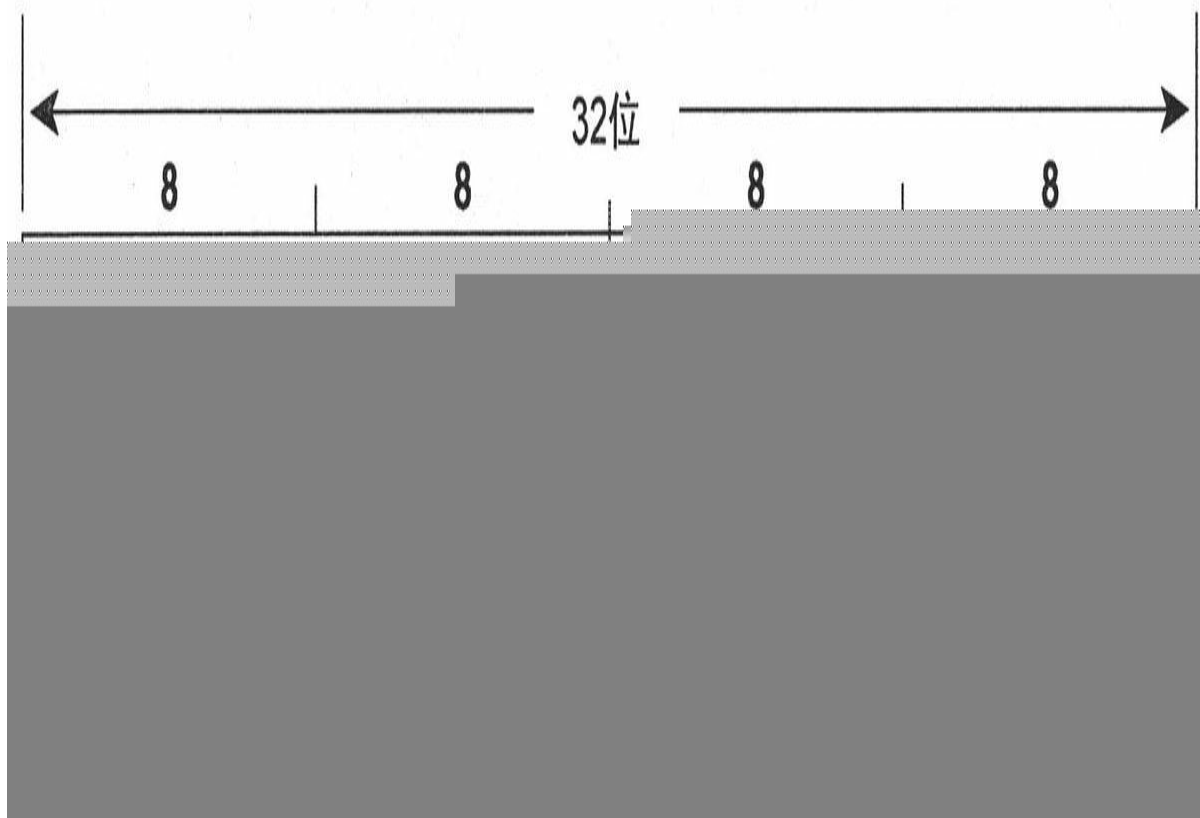


图6-10 RIPng的消息格式

- 命令 ——总是设置为1或2，1表示本消息是请求消息，2表示本消息是响应消息。
- 版本号 ——总是设置为1，RIPng当前的版本是RIPngv1。
- IPv6前缀 ——是指路由条目的128位的目的IPv6前缀。
- 路由标记 ——与RIPv2中16位的路由标记字段的用法相同：用来标记经过RIP路由域传送的外部路由属性。
- 前缀长度 ——是一个8位的字段，用来指出IPv6前缀字段中的地址的有效位数。例如，假如通告的前缀是3ffe:2100:1201: : /48，那么它的前缀长度字段的值是48（0x30）。如果所通告的是缺省路由，它的IPv6前缀是0:0:0:0:0:0:0:0，前缀长度是0。
- 度量 ——与RIPv1和RIPv2中一样，是指跳数度量。但是当最大可能值为16时，这个字段会把RIPv1和RIPv2里不必要的16位大小的

位数减小为8位。

RIPng使用和RIPv2相同的方式指定下一跳地址。换句话说，如果是一个有效的非零下一跳地址，表示下一跳路由器不同于发起响应消息的路由器；如果下一跳地址是全0（0:0:0:0:0:0:0:0），则表示下一跳路由器就是发起响应消息的路由器本身。但是，与RIPv2中每一个路由条目都跟随一个下一跳地址不同，RIPng中只在一个专门的路由条目里指定下一跳地址，并把所有使用这个下一跳地址的路由条目编成组，跟在这个专门的路由条目后面。也就是说，下一跳路由条目指定的下一跳地址应用于跟随在它后面的所有路由条目，要么到这个响应消息的末尾，要么直到发现另一个专门的下一跳路由条目。

如图6-11所示，图中显示了下一跳路由条目的格式。它的128位地址要么是另一台路由器的IPv6地址，要么是：：——表示发起该消息的路由器的地址。路由标记和前缀长度字段都设置为全0。由于接收路由器的度量字段设置为全1（0xFF），因此它可以识别这是一个下一跳路由条目。



图6-11 设置为全1的度量值表示该路由条目是一个下一跳路由条目。所有跟在这个条目后的路由条目都使用这个下一跳地址，除非是在该消息的末尾或者出现另一个下一跳路由条目

6.3 RIPv2的配置

由于RIPv2是RIPv1的增强版，而不是一个单独的协议，因此，在第5章中介绍的某些命令都可以以同样的方式在RIPv2中正确使用，例如计时器和度量的操作、单播更新或根本不发出更新的配置等。当浏览RIPv2协议的配置后，本节余下来的部分将集中讲述一些新的扩展特性的配置。

6.3.1 案例研究：RIPv2的基本配置

缺省时，在Cisco路由器上配置一个RIP进程将只发送RIPv1的消息，但是同时接收RIPv1和RIPv2的消息。这个缺省的配置可以通过命令`version`来更改，参见示例6-1。

示例6-1 命令`version 2`使RIP只能接收和发送RIPv2的消息



在这种配置方式下，路由器只发送和接收RIPv2的消息。同样地，路由器也可以配置成只发送和接收RIPv1消息的方式，参见示例6-2。

示例6-2 命令`version 1`使RIP只能接收和发送RIPv1的消息



可以在路由器配置模式（`config-router mode`）下输入命令`no version`恢复到原来的缺省方式。

6.3.2 案例研究：与RIPv1的兼容性

RFC 2453中建议的基于接口模式下的“兼容性开关”，在Cisco IOS软件中

可以通过命令**ip rip send version** 和**ip rip receive version** 来实现。

在图6-12所示的网络中包含了同时宣告RIPv1和RIPv2的路由器。另外，主机Pojoaque是一台运行“routed”进程的Linux主机，它只能理解和处理RIPv1。路由器Taos的配置参见示例6-3。

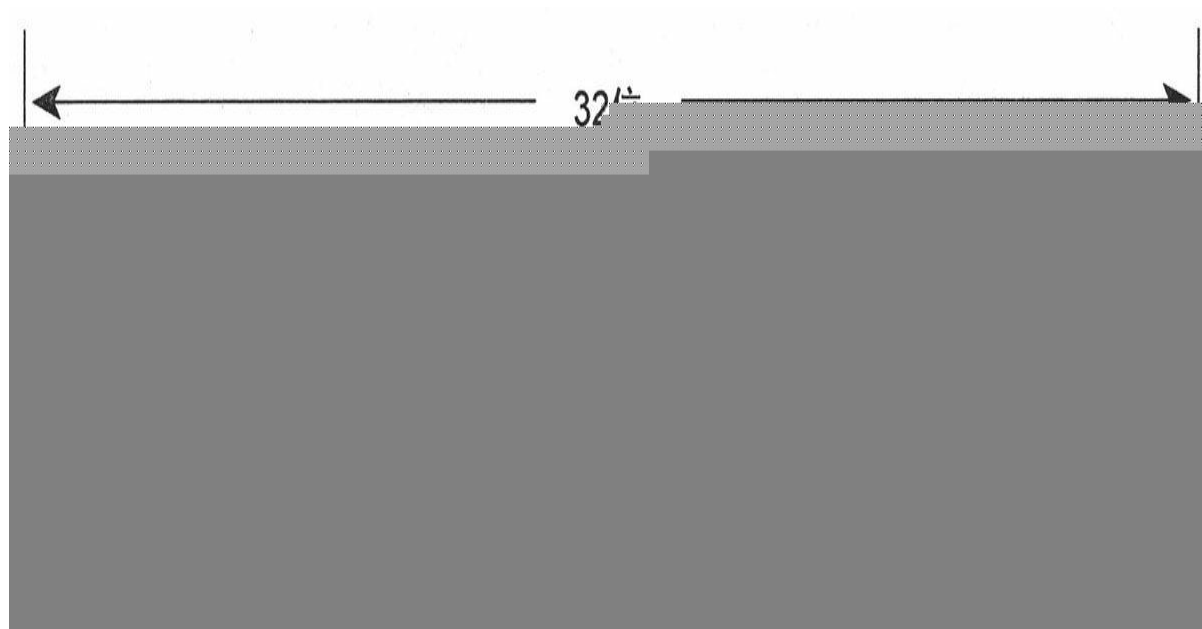
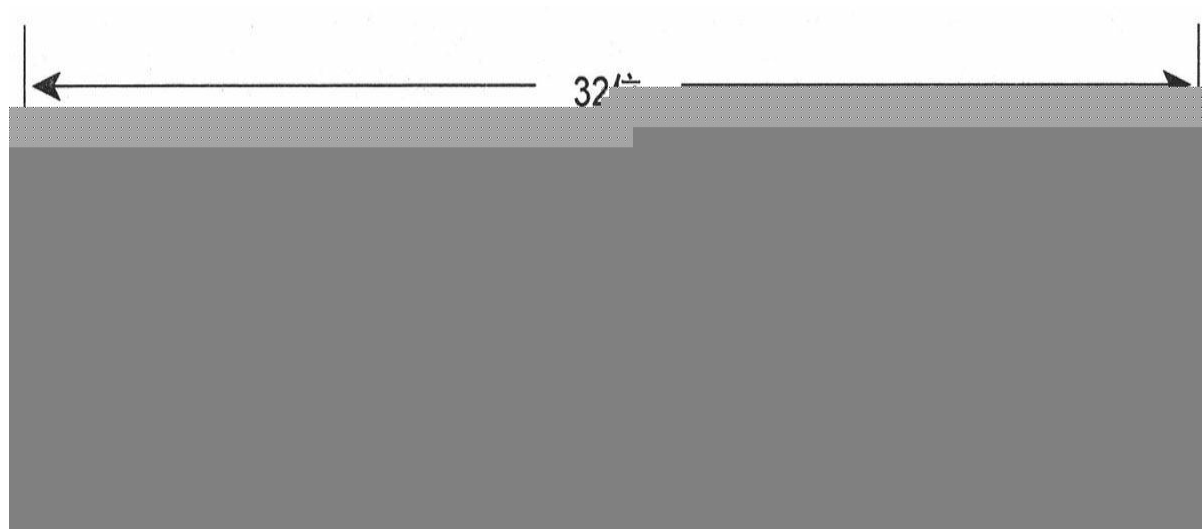


图6-12 路由器**Taos**正在运行**RIPv2**，但是必须宣告版本**1**给其他一些设备

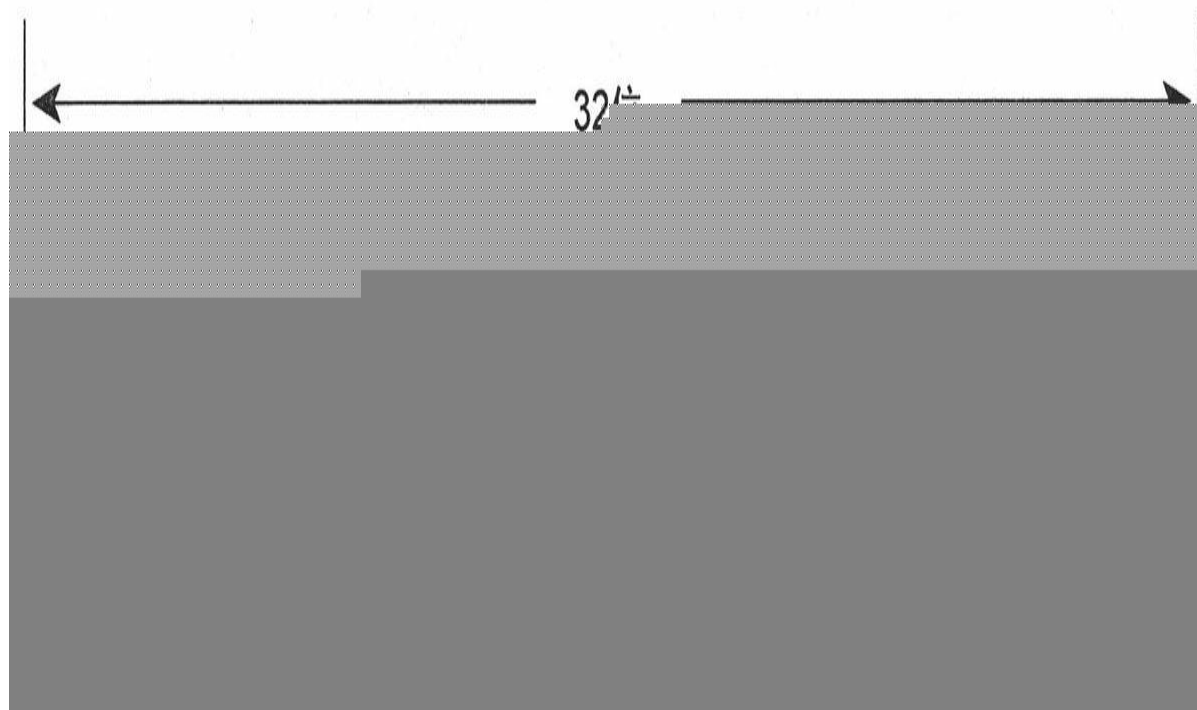
示例6-3 路由器**Taos**的配置



因为路由器Laguna是RIPv1的一个宣告者，所以路由器Taos的E0接口要配置成接收和发送RIPv1更新的方式，而E1接口则配置成同时支持版本1和版本2更新的方式，以便发送给运行RIPv1的路由器Pojoaque和运行RIPv2的路由器Sandia。E2接口没有什么特别的配置，它将按照缺省方式发送和接收版本2的更新。

在示例6-4中，使用**debug ip rip** 命令观察路由器Taos发送和接收的消息。这里有几个有趣的地方。首先，注意观察RIPv1和RIPv2消息携带的内容不同。RIPv2的更新中可以看到地址掩码、下一跳和路由标记字段（在这个实例中，都被设置成全0）。其次，可以观察到，E1接口正在以广播方式发送RIPv1更新，而以多播方式发送RIPv2更新。第三，由于路由器Taos没有配置接收RIPv1，从而来自路由器Pojoaque（172.25.150.206）的更新将被忽略（路由器Pojoaque被错误地配置，并且正在广播它的路由表）。[\[9\]](#)

示例6-4 使用调试功能，可以从路由器**Taos**上看到**RIP**版本的发送和接收



在示例6-4中，可能最需要关注的就是广播到主机Pojoaque的更新了，它没有包含子网 172.25.150.32。路由器Taos是通过多播方式的RIPv2更新从路由器Sandia学习这个子网的。但是主机Pojoaque由于只宣告RIPv1而

不能接收这些多播。此外，虽然路由器Taos得知这个子网，但是水平分隔法则禁止路由器Taos把从这个接口学到的路由再从相同的接口通告出去。

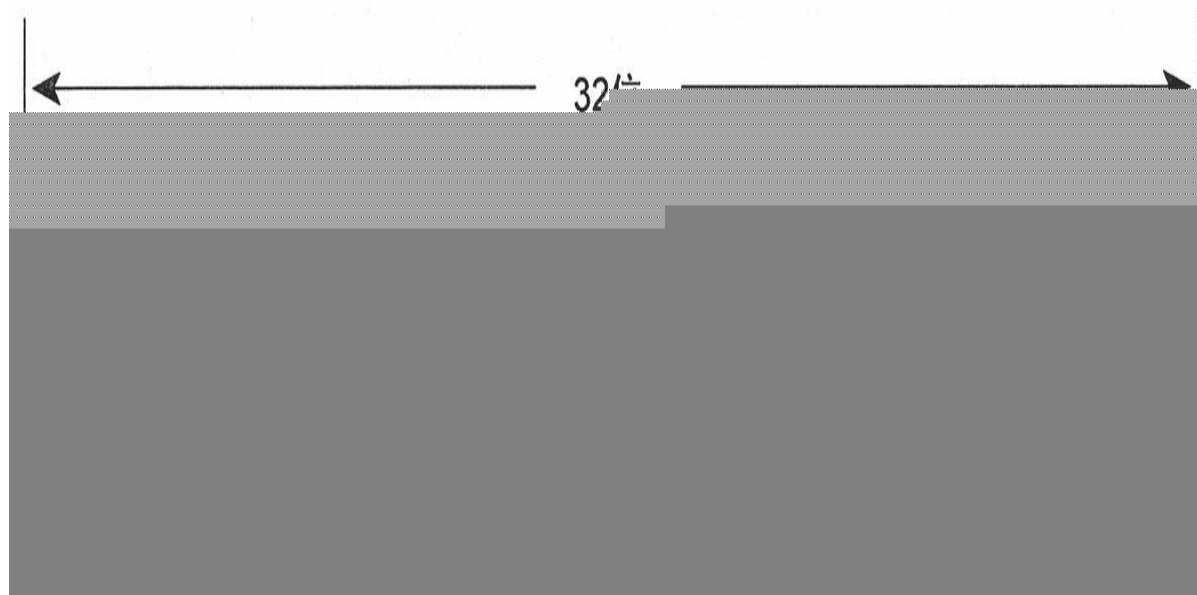
因此，主机Pojoaque无法得知子网172.25.150.32。这里有可供使用的两种修正方法：第一，把路由器Sandia配置成可以同时发送RIP协议的两个版本；第二，可以通过示例6-5的配置在路由器Taos的E1接口上关闭水平分隔。

示例6-5 在路由器Taos的配置中关闭了水平分割



示例6-6中显示了更改配置后的结果。现在路由器Taos在它的更新里包含了子网172.25.150.32。可以预见到关闭水平分隔后的一些可能的结果：路由器Taos现在不仅通告子网172.25.150.32给主机Pojoaque，而且把这个子网通告回路由器Sandia。

示例6-6 在E1接口上关闭水平分隔后，路由器Taos在通告给主机Pojoaque的更新中包含了子网172.25.150.32

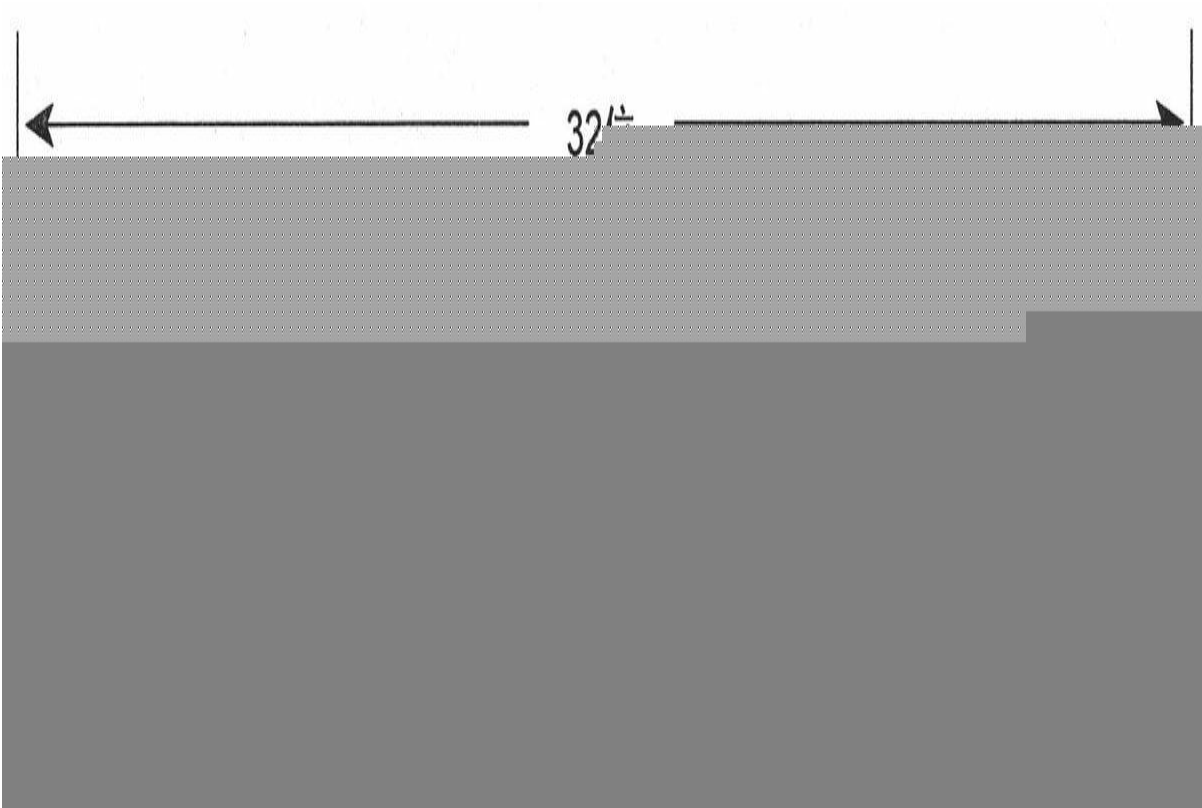


6.3.3 案例研究：使用VLSM

参见图6-12，子网172.25.150.0/24已经分配给图中的网络，这个子网通过扩展到28位的掩码进一步子网化以满足不同的数据链路。表6-2中显示了以二进制和点分十进制表示的可用子网。根据公式 $2^n - 2$ ，每一个子网[10]可以含有14个主机地址。在这些子网中，172.25.150.32、172.25.150.192和172.25.150.224已经被使用。



表6-2 VLSM应用于子网172.25.150.0/24



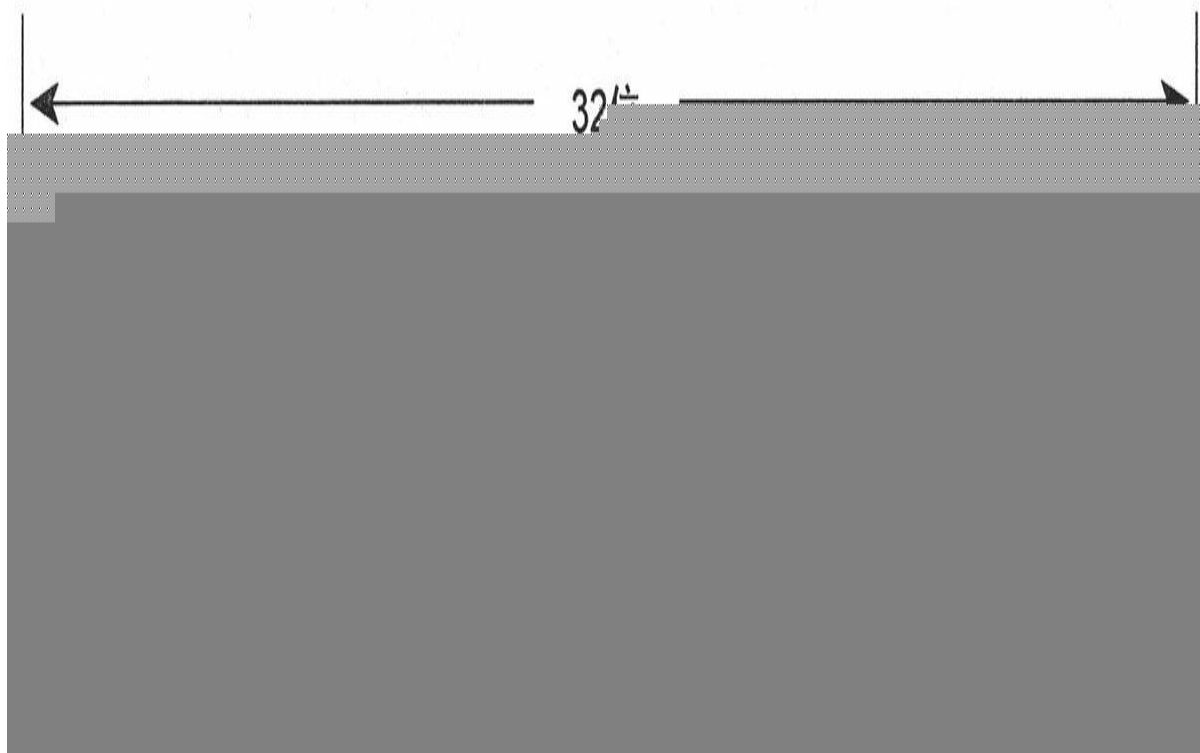


图6-13 VLSM技术可以用来满足单个数据链路的地址需求

在图6-13中，路由器Taos增加了一个新的以太网段Ethernet3，它包含了60台主机。为了给这个数据链路分配地址，需要至少有6位主机位的子网。有类别路由选择协议如果使用次一级的地址，应该需要分配5个表6-2中划分的子网给以太网段Ethernet3[$5 \times (2^4 - 2) = 70$]。利用无类别路由选择协议和VLSM技术，表6-2中的4个子网可以汇总成一个单一的带有26位掩码的子网。这一个步骤可以提供6个主机位（62个主机地址），并且没有必要使用次一级的编址。这里的4个子网172.25.150.64/28~172.25.150.112/28可以合并成单个26位掩码的子网172.25.150.64/26。注意，这里的4个子网并不是随意选择的，在上面的16个子网组合中，它们开始的26位掩码必须是相同和惟一的。[\[11\]](#)

参见图6-13，图中网络增加了4台路由器和4条串行链路。如果没有使用VLSM技术，这4条串行链路就需要使用表6-2中的4个子网；而使用VLSM技术，只要使用表6-2中的1个子网就可以满足所有4条串行链路的需求了。选用子网172.25.150.240，利用30位的掩码创建表6-3中的子网，这4个更小的子网的任何一个都包含两个主机地址。

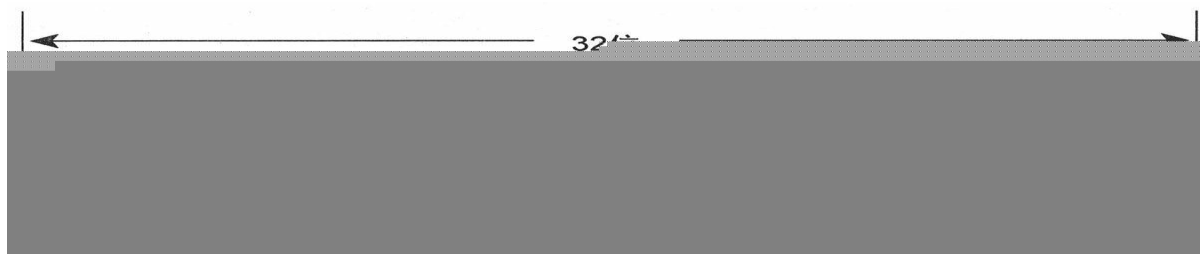


表6-3 应用30位的掩码对子网172.25.150.240再进
行子网划分

划分子网的基本目的总是相同的：路由器必须能够使用惟一的地址来标识每一条数据链路，以区别于网络中的其他地址，这也是前面两个例子的共同目的。在第一个例子中，通过减小掩码的大小，将多个地址合并成一个地址，该操作一直进行到所有地址都具有相同的位。注意，这种情况在子网被汇总成主网络地址时也会发生。在第二个例子中，通过扩展子网掩码将单个子网划分为多个更小的子网。

6.3.4 案例研究：不连续的子网和无类别路由选择

图6-14显示出新添了4台路由器，并且每一台路由器都连接两个以太网段。其中每一处都包含一个属于子网172.25.150.0/24的以太网段，并且都不超过12台主机。这个需求很容易满足，从表6-2中选用4个未用的子网进行分配即可。

每台新添路由器下挂的另一个以太网属于子网192.168.50.0，并且都不超过25台主机。子网192.168.50.64/26和192.168.50.128/26正在被其他链路使用，只剩下子网192.168.50.0/26和192.168.50.192/26。通过增加掩码位到27位，这两个子网就被划分成了4个子网，每个子网有5个主机位——每个子网可以提供30个主机地址。表6-4显示了这4个子网的二进制表示。



表6-4 使用27位掩码对子网192.169.50.0/26进一步

划分

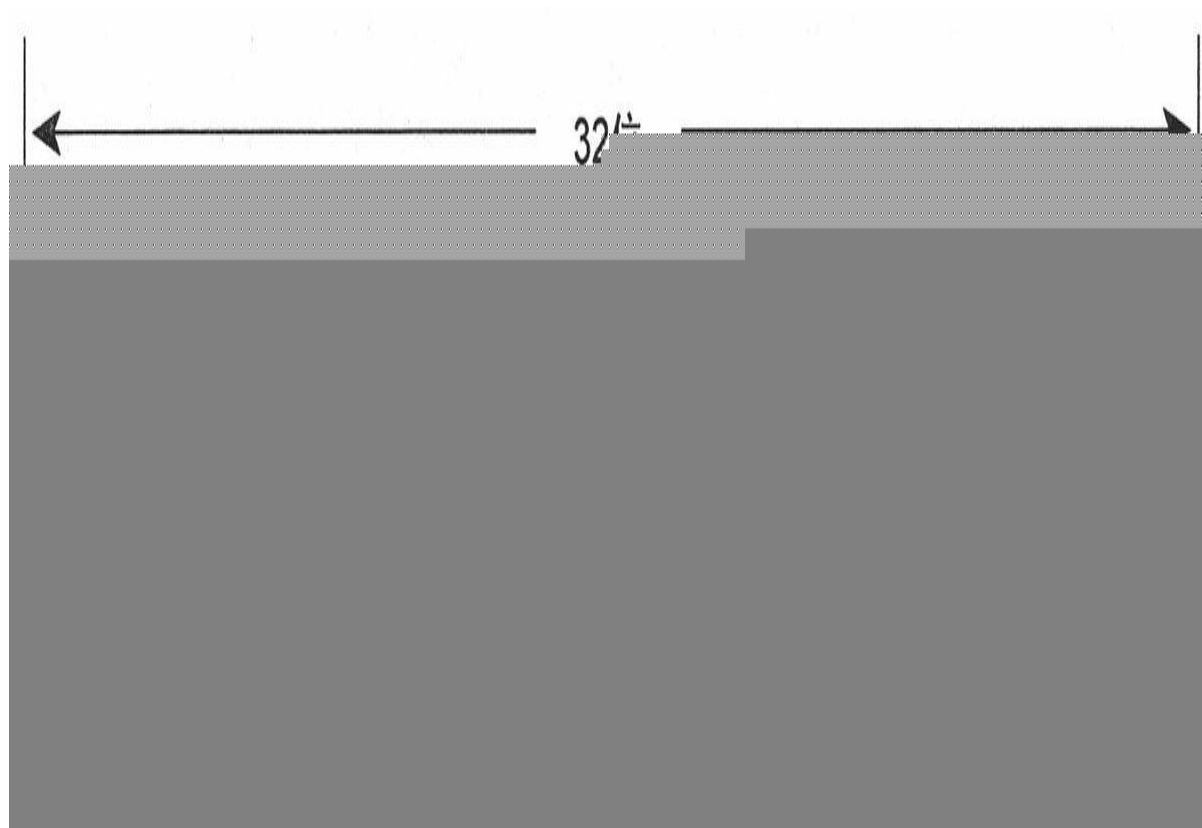
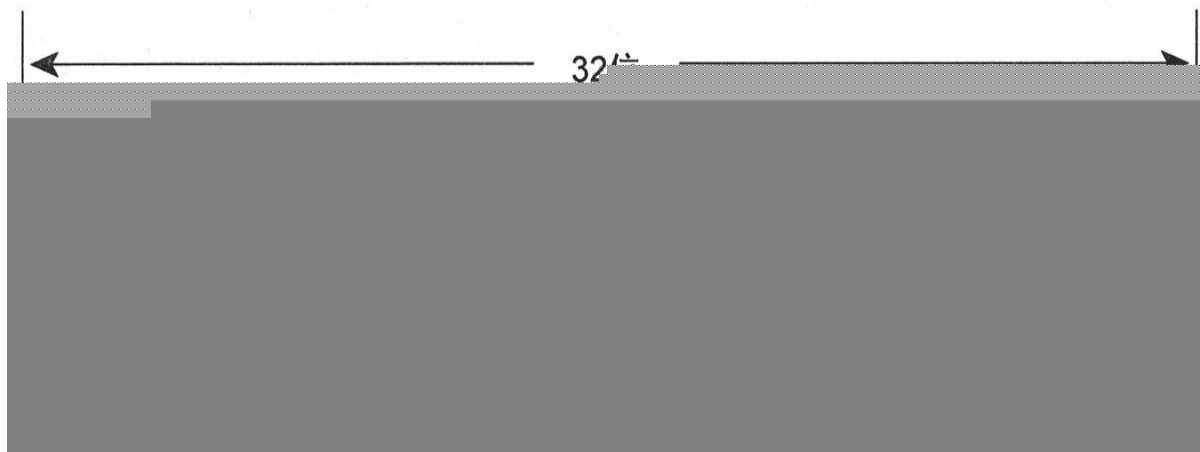


图6-14 路由器**Cochiti**、**Isleta**、**Jemez**和**Tesuque**中的每一台连接了两个以太网段。每台路由器上都包含一个属于子网**172.25.150.0/24**的以太网，另一个则属于子网**192.168.50.0/24**

分配完所有的子网地址后，下一个需关注的就是这样一个事实：**192.168.50.0**的子网是不连续的。第5章展示了一个不连续子网的实例，并演示了怎样使用辅助接口连接它们的。而无类别路由选择协议没有关于不连续子网的这些困难。因为每一条路由更新都包含一个子网掩码，因而一个主网络的子网能够通告给其他的主网络。

但是，**RIPv2**协议在缺省情况下会在主网络边界上进行路由汇总，这一点和**RIPv1**相同。为了关闭路由汇总功能以允许被通告的子网通过主网络边界，可以在**RIP**的处理中使用**no auto-summary** 命令。路由器**Cochiti**的配置参见示例6-7。

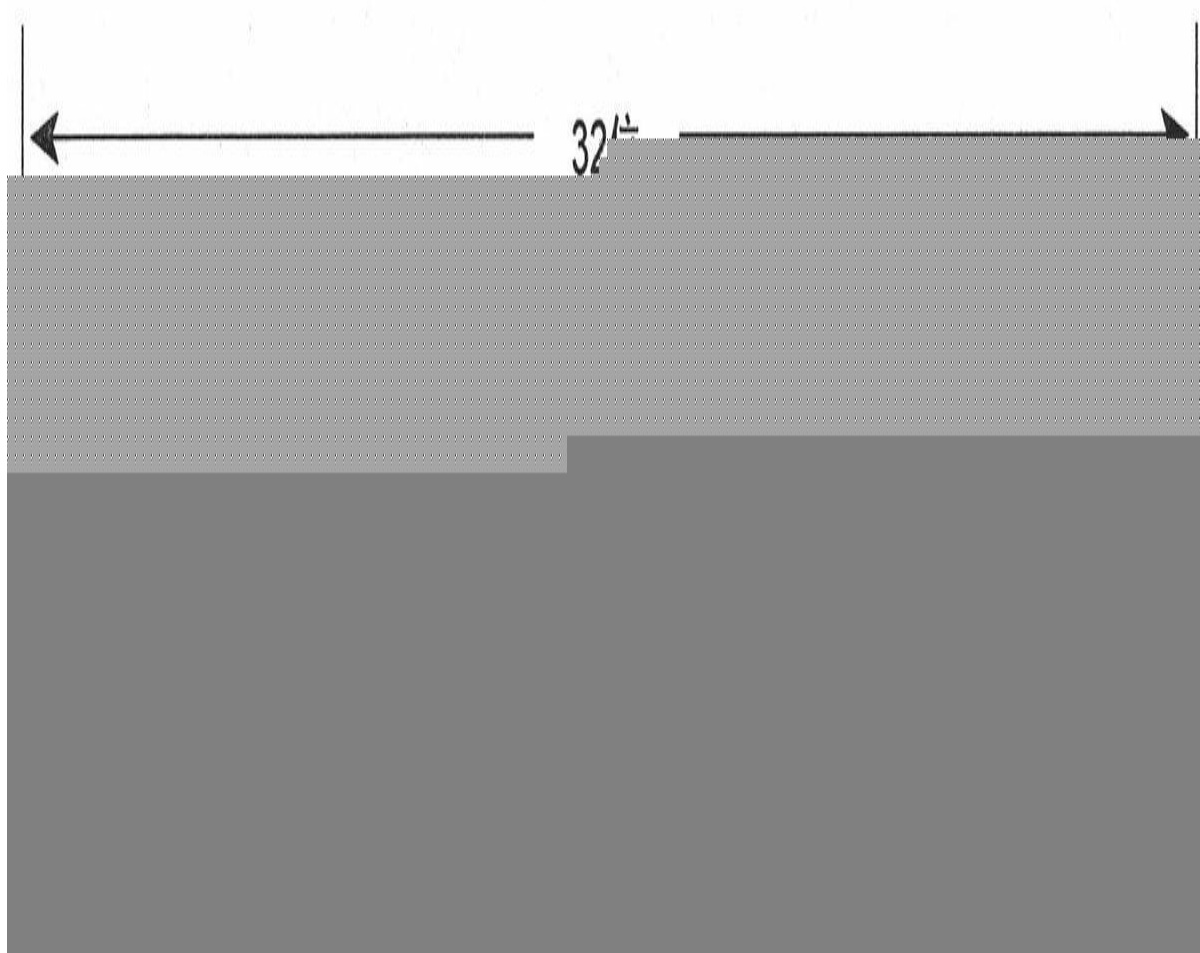
示例6-7 路由器**Cochiti**的配置，不再支持路由自动汇总



路由器Isleta、Jemez和Tesuque也可以进行类似的配置。在路由器Taos和Acoma上也必须要关闭路由汇总。回忆图6-12中，路由器Laguna运行的版本是RIPv1，为了使这个配置能够正常工作，必须将RIP的版本更改为版本2。

读者仔细考虑一下，可变长子网掩码对仍旧运行RIPv1的主机Pojoaque产生了什么影响。示例6-8中的调试信息显示路由器Taos发送到子网172.25.150.192/28上的版本1和版本2的更新消息。版本1的更新消息仅仅包含那些28位掩码的子网，这与正在广播更新的子网的掩码是一样的。虽然主机Pojoaque不接收子网172.25.150.64/26或所有串行链路的子网的通告，但是在这个实例中，关于那些子网地址的分析显示，主机Pojoaque依然可以正确地识别出这些子网与它本身的子网地址不同。到达这些子网的数据包也会送到路由器Taos上进行路由选择。

示例6-8 虽然来自路由器Taos的RIPv2更新包含了图中网络的所有子网，但是RIPv1的更新消息却仅仅包含到达网络192.168.50.0的汇总路由和网络172.25.150.0的一些子网（这些子网的掩码和发送这些更新的接口掩码相同）



6.3.5 案例研究：认证

Cisco实现RIPv2的消息认证包含了两种选择——简单的口令或MD5认证。另外，也包含了在一个“钥匙链”上定义多个钥匙或口令的选项。这样路由器就可以在不同的时候配置不同的钥匙。

设置RIPv2认证的步骤如下：

步骤1： 定义一个带名字的钥匙链。

步骤2： 定义在钥匙链上的钥匙。

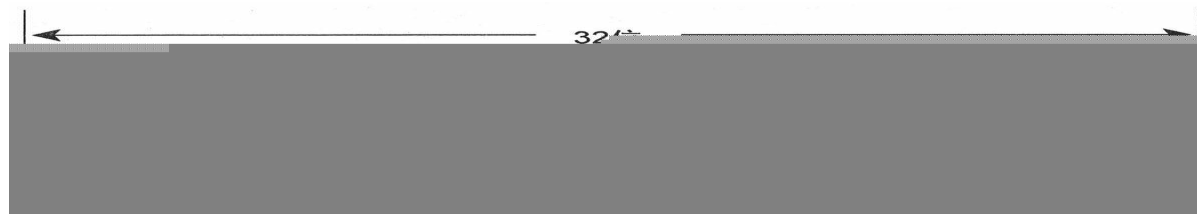
步骤3： 在接口上启动认证并指定使用的钥匙链。

步骤4： 指定这个接口使用明文认证还是MD5认证。

步骤5： 可选地配置钥匙的管理。

在示例6-9中，路由器Taos上配置了一个名为“Tewa”的钥匙链，这个钥匙链上惟一的一个钥匙是“Key 1”，它含有一个口令“Kachina”。E0接口将使用MD5认证的这个钥匙去验证来自路由器Laguna的更新消息。

示例6-9 路由器Taos有关认证的配置

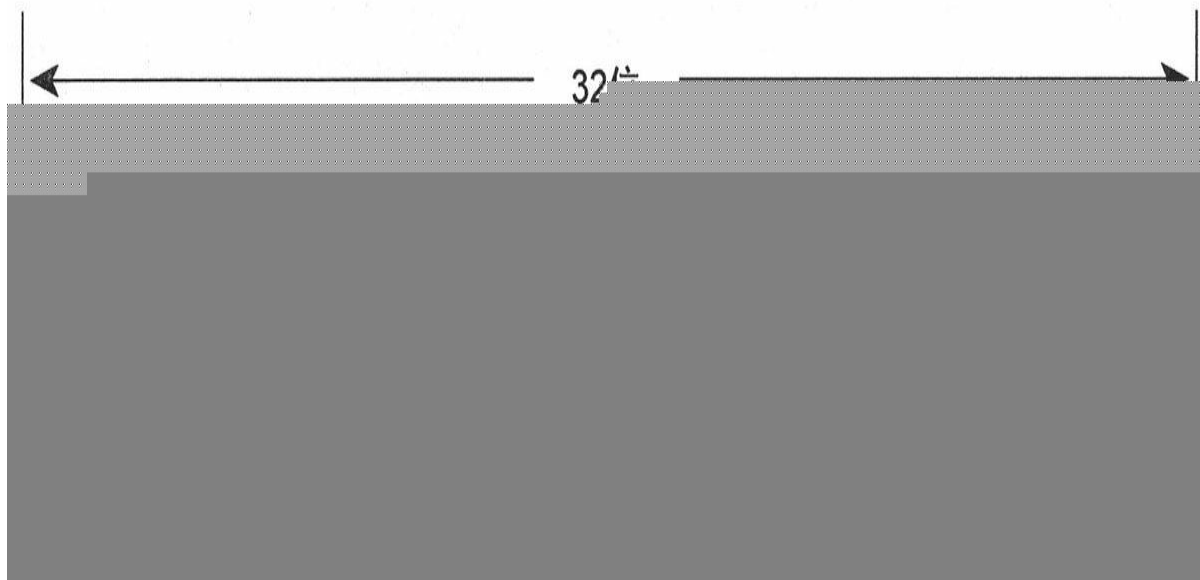


即使只有一个钥匙，也必须配置钥匙链。虽然进行带认证的更新消息交换的所有路由器必须拥有相同的口令，但是钥匙链的名字却只在本地路由器上有意义。例如，路由器Laguna可以有一个名为Keres的钥匙链，但是宣告给路由器Taos的钥匙的字符串必须是Kachina。

如果没有添加命令**ip rip authentication mode md5**，接口将使用缺省的明文认证。虽然在与一些RIPv2的设备通信时明文认证可能是必要的，但是只要有可能，几乎总是明智地使用安全性能好得多的MD5认证。

钥匙管理（key management）用来进行从一个认证钥匙到另一个认证钥匙的迁移工作的。在示例6-10中，路由器Laguna的配置是：在2004年7月1日下午4:30开始使用第一个钥匙，使用的时长是12h（43200s）；第二个钥匙从2004年7月2日凌晨4:00开始生效，并一直使用到2004年12月31日下午1:00；第三个钥匙从2004年12月31日下午12:30开始生效，并在这个时间以后永久有效。

示例6-10 路由器Laguna有关认证的配置



正如配置所显示的，从其他路由器接受的口令和发送消息所使用的口令在管理上是分离的。因此，使用命令**accept-lifetime** 和**send-lifetime** 都应该含有一个指定的开始时间和一个指定的持续时间或结束时间，或者指定关键字 ***infinite***。钥匙的号码按照从最低到最高的顺序检查，使用第一个有效的钥匙。

虽然这个配置可以使用30min的时间重叠来在不同的系统时钟之间进行校正，但是，这里强烈建议在对钥匙的管理时，使用像网络时钟协议（Network Time Protocol,NTP）这样的时钟同步协议（Time Synchronization Protocol）。 [\[12\]](#)

6.4 RIPng的配置

RIPng的配置步骤和RIPv1与RIPv2是相同的：首先创建一个路由选择进程，接着在接口上启动该路由选择协议，然后执行任何所希望的用户配置。但是，RIPng中所使用的命令却是非常不同。事实上，路由选择进程的创建和第一个接口上路由选择协议的启动都是使用一个接口命令一步完成的。RIPng的案例研究与RIPv1和RIPv2中演示的那样，处理过程是相似的，但配置是不同的。

6.4.1 案例研究：RIPng的基本配置

在路由器上启动IPv6的单播路由选择协议后，只需要一条命令就可以启动IPv6路由选择的RIPng协议。所需命令是一个接口模式下的命令：**ipv6 rip process-name enable**，它可以创建一个名字为process-name的RIPng进程，同时在接口上启动RIPng协议。该命令需要在每一个希望运行RIPng协议的接口上启动。

如图6-15所示，图中显示了IPv6、RIPng和在图6-14的网络基础上增加的一些新的以太网接口。



图6-15 在路由器Laguna、Taos和Sandia的接口上配置的IPv6地址运行RIPng协议

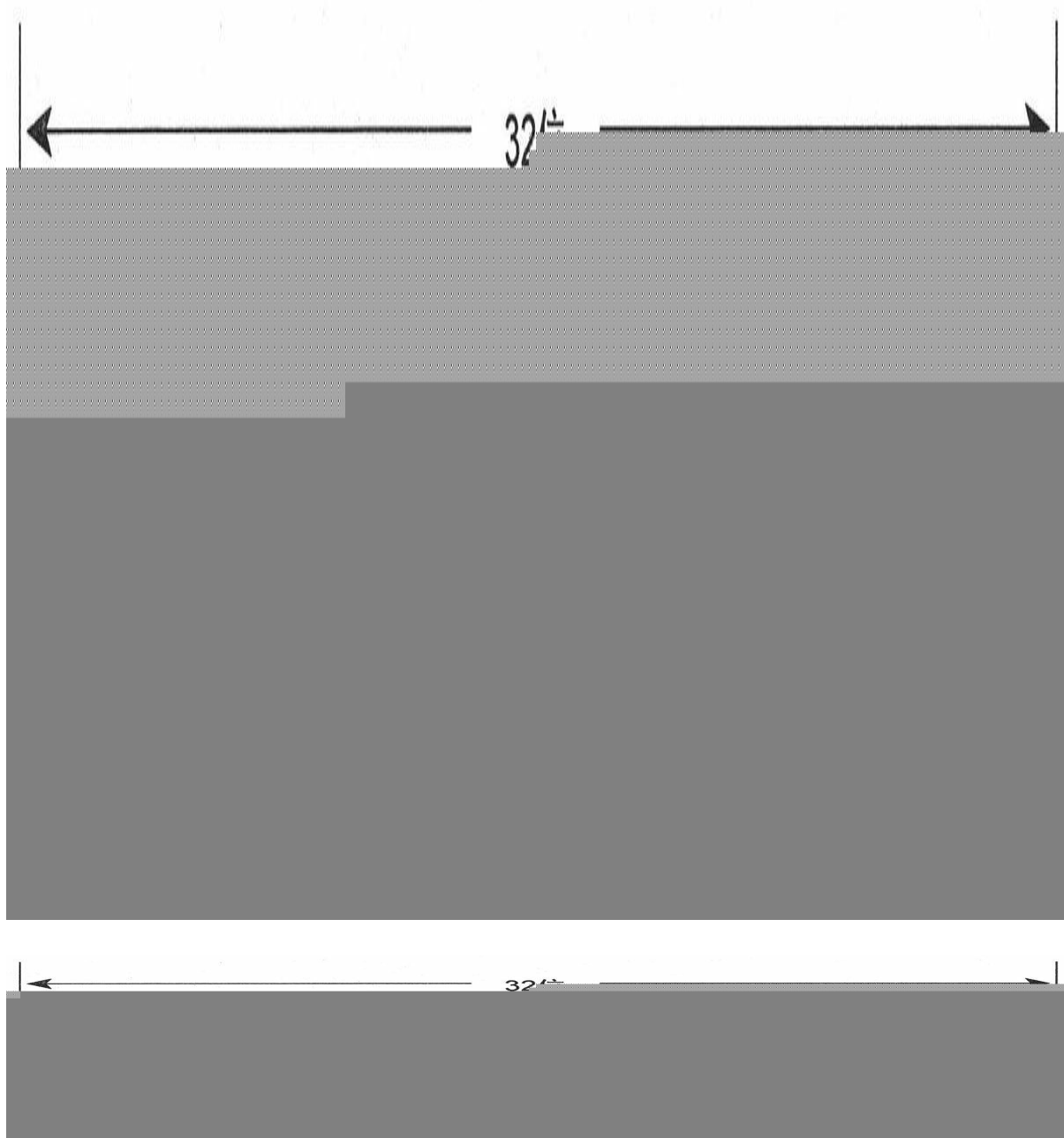
示例6-11 中显示了路由器Taos的配置。示例6-11 路由器Taos的IPv6和RIPng配置



RIPng的创建和在接口上激活RIPng都是通过一个接口配置命令实现的。这是RIPng的配置与RIPv1和RIPv2的配置之间的不同之处。回忆一下，在RIPv1和RIPv2的配置中需要一个全局命令（**router rip**）来创建进程，还需要**network <address>**语句指定运行RIP进程的接口。在RIPng中，只需要在第一个运行RIPng的接口上配置接口命令**ipv6 rip bigMountain enable**，一个命名为bigMountain的RIPng进程就自动地创建了。另外，IOS可以自动地创建一个路由选择进程的配置条目（**ipv6 router rip bigMountain**）。

路由器Taos上RIPng进程的名字指定为bigMountain。这里请注意，进程的名字是在每一个接口上指定的。这个名字只是和本地路由器相关联，用来区分可能运行在同一台路由器的多个RIPng进程。每一个不同接口都可以运行一个惟一的RIPng进程，而在这些接口之间不进行信息交换，或者在单个接口上运行多个进程。另一方面，在同一台路由器上只能运行单个RIPv1或RIPv2进程，一个接口要么属于这个进程，要么不运行RIP协议。[\[13\]](#)

示例6-12 路由器Taos的IPv6路由表



如图6-16所示，在路由器Acoma上增加了两个新的以太网接口。新的网络策略是，在图6-16的网络上，IPv6通信流量应该根据以下策略进行分段：与路由器Sandia相连的LAN上的主机之间可以互相访问，同时它们也可以访问路由器Acoma连接的Ethernet1上的设备。与路由器Laguna相连的主机可以访问路由器Acoma连接的Ethernet2上的设备。路由器Sandia与Laguna上的主机不需要互相访问。路由器Acoma连接的Ethernet1与Ethernet2之间不需要相互访问。

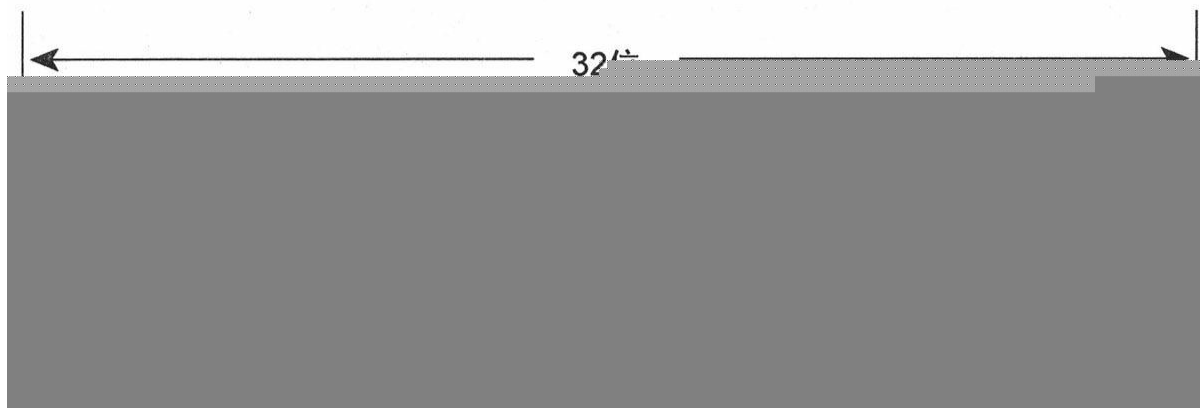
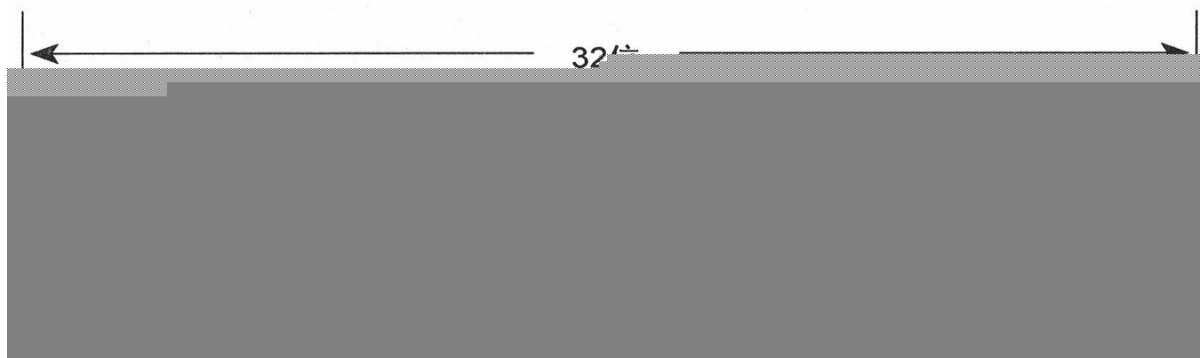


图6-16 路由器Acoma增加到上述网络中，它带有两个运行IPv6的以太网客户LAN网段

示例6-13中的配置被加入到路由器Taos中。

示例6-13 路由器Taos的配置通过创建多个RIPng路由选择进程来支持对IPv6网络流量进行分段的网络策略



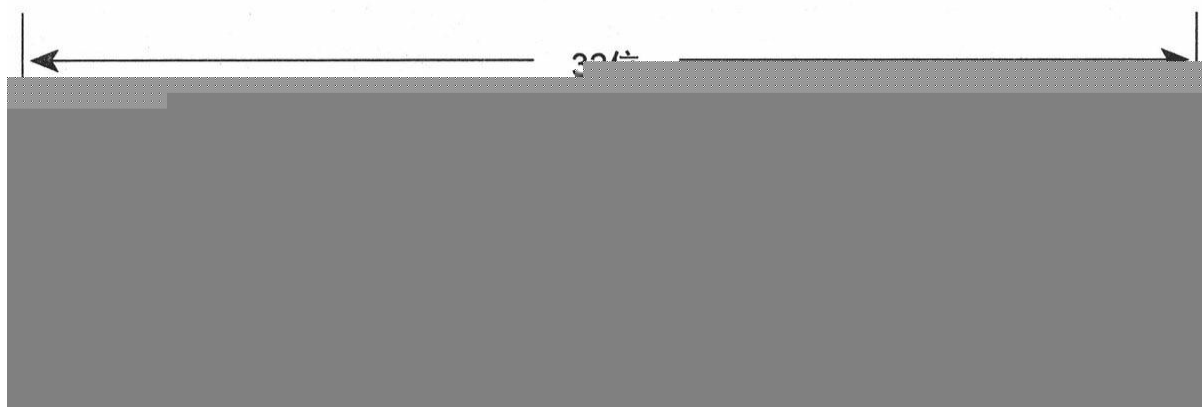
连接到路由器Laguna的接口Ethemet0仍然属于名字为bigMountain的RIPng进程。在连接到路由器Sandia的接口Ethernet1上启动了一个新的名字为smallMountain的进程，并在这个接口上关闭了bigMountain进程。最后在连接到路由器Acoma的接口Ethemet2上同时启动了这两个进程。如果在同一个接口上同时运行两个进程，那么这两个RIPng路由选择进程不能使用相同的UDP端口号。通过改变smallMountain进程的端口号，路由器Taos发送smallMountain进程的更新到UDP端口527上，而bigMountain进程则仍然使用缺省的RIPng端口号（即UDP端口521）发送更新消息。接收路由器可以使用端口号来区分所收到的更新消息属于哪一个路由选择进程。一台运行单个路由选择进程的RIPng路由器为了处理每一个更新消息，它将使用相同的UDP端口号接收来自一台路由器

上运行的多个RIPng进程发送的更新消息。如果这两个更新消息有关同一个地址的信息存在冲突，例如度量值不同，那么接收路由表可能会根据每一个更新消息改变。如果一台接收路由器上正在运行多个RIPng进程，并且使用相同的UDP端口号，那么这台接收路由器将不能区分哪一个RIPng进程应该接收这个更新消息。改变第二个和随后的RIPng进程的UDP端口号可以排除这些故障。所有运行更改UDP端口号的RIPng进程的路由器和主机都必须使用相同的UDP端口号。由于路由器和其他TCP/IP设备使用端口号识别数据包需要哪一个UDP进程进行转发，而RIPng更新消息以多播方式向所有的RIPng路由器发送，因此新的UDP端口号一定不能与任何运行RIPng的路由器上任何其他进程的端口号相同。在这个例子中为smallMountain进程配置的端口号是527。和smallMountain命名的RIPng进程交换RIPng更新消息的路由器必须更改它的端口号。路由器Sandia和Acoma的配置更改分别参见下面的示例6-14与示例6-15。

示例6-14 在路由器**Sandia**上更改UDP端口号的RIPng配置



示例6-15 在路由器**Acoma**上更改UDP端口号的RIPng配置

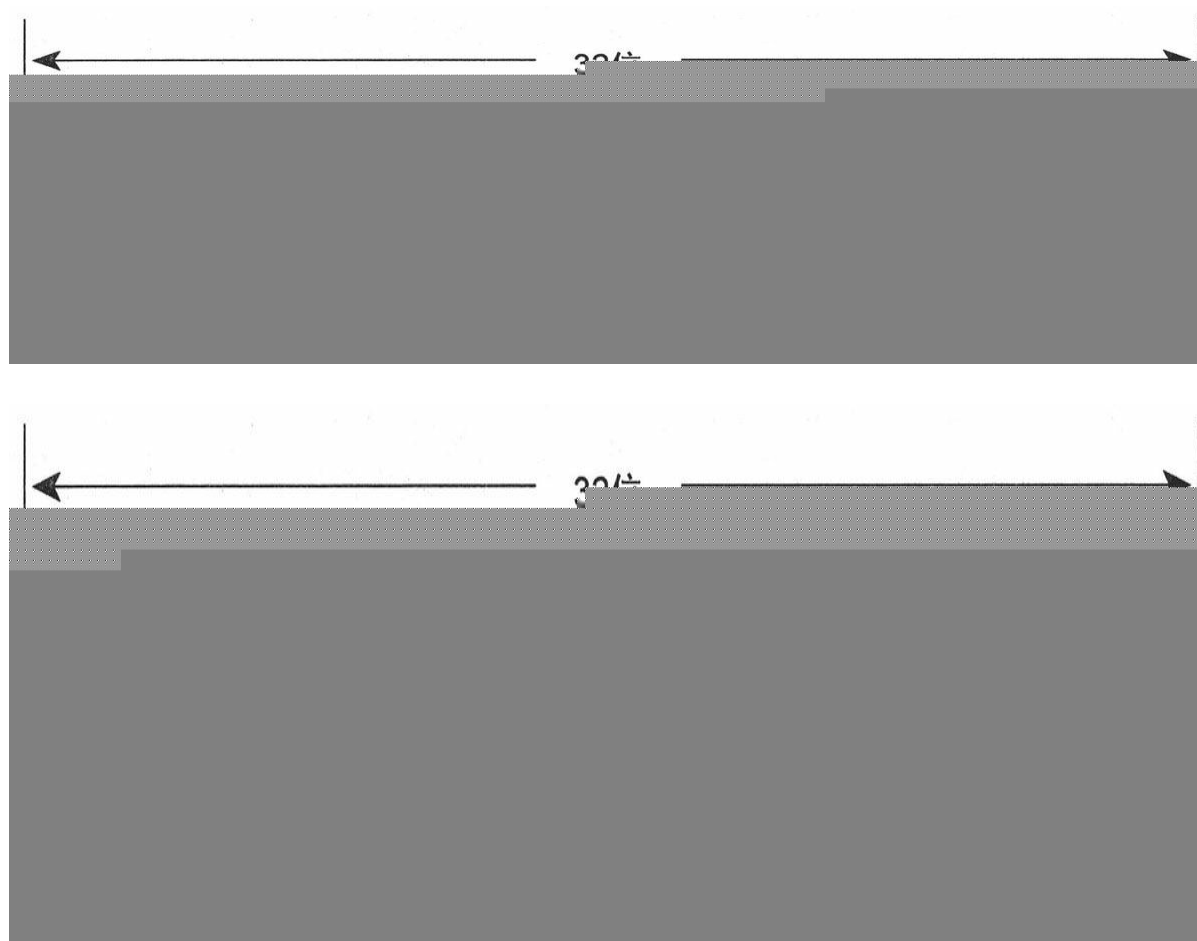


读者在这里可以注意到，路由器Acoma上的进程名字和路由器Taos上的进程名字是不同的。RIPng进程的名字只对本台路由器有意义，它们并不在路由更新之间进行交换。路由器Acoma的以太网接口Ethemet0连接到路由器Taos的以太网接口Ethemet2上。它运行了两个路由选择进程：hill和summit。进程hill也运行在Ethemet1的接口上。Ethemet1接口的IPv6地址使用更改后的UDP端口号527通告给路由器Taos。这是与路由器Taos上

的smallMountain进程相关联的。因此，IPv6的地址将通告给路由器Sandia，而不是Laguna。

示例6-16中显示了运行在路由器Taos上的RIPng路由选择进程的相关信息。使用命令**show ipv6 rip**可以显示出每一个进程的信息。注意并行路径缺省的最大数目是16，而计时器的缺省值还没有更改。

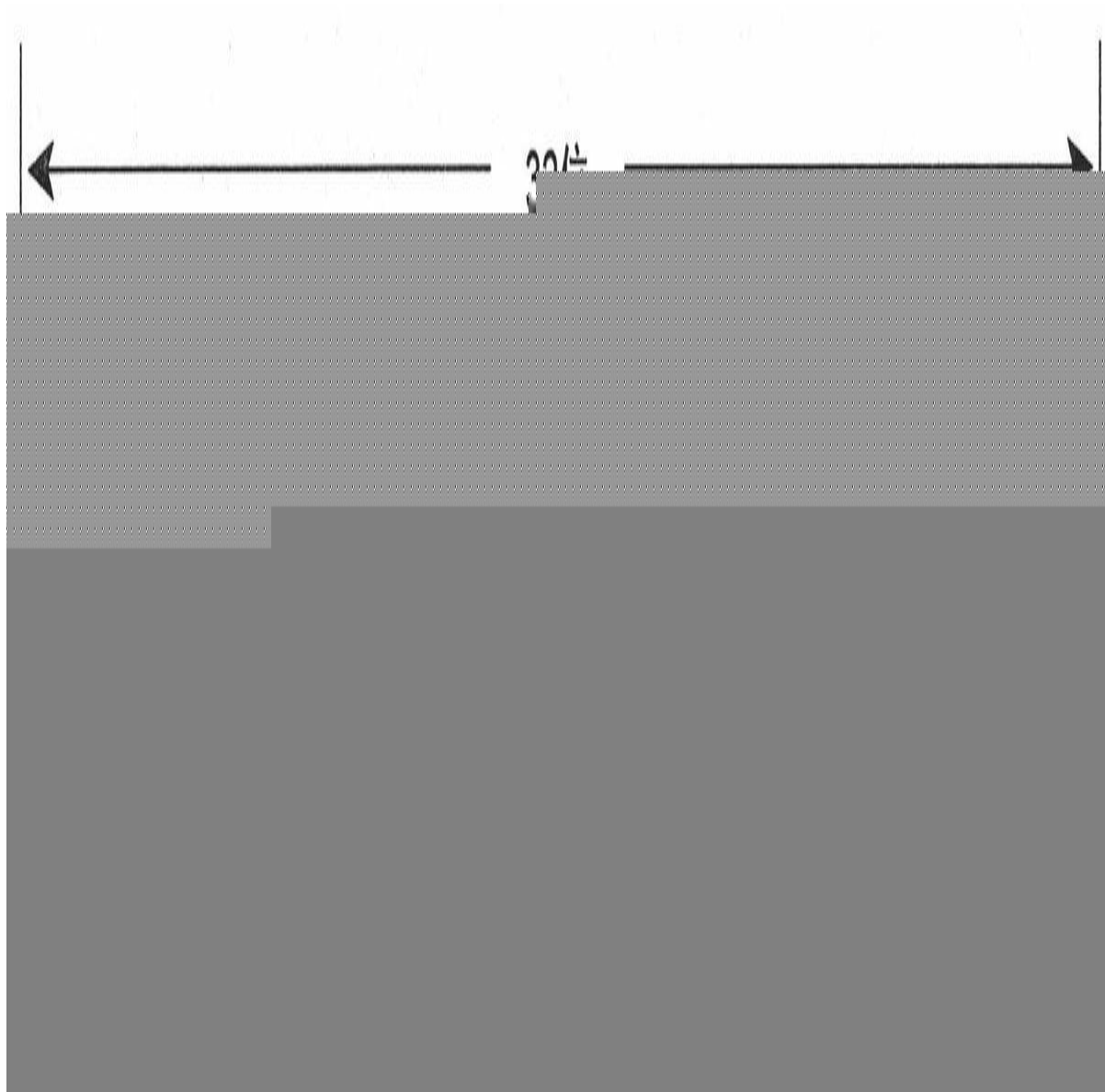
示例**6-16** 使用命令**show ipv6 rip**可以显示出运行在某台路由器上的每一个**RIPng**进程的相关信息



进程bigMountain运行在接口Ethemet0与Ethemet2上。接口Ethernet1和Ethemet2同属于进程smallMountain，并且共同使用UDP端口号527。

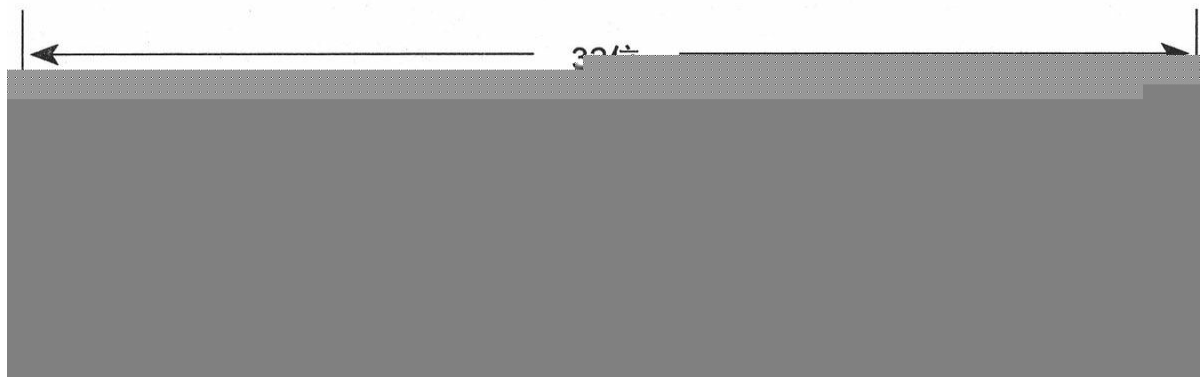
IPv6路由表显示出路由器Taos已经把这两个进程的路由加载到它的路由表中了，参见示例6-17。

示例6-17 IPv6路由表显示了来自每一个运行的RIPng进程的所有学习到的路由

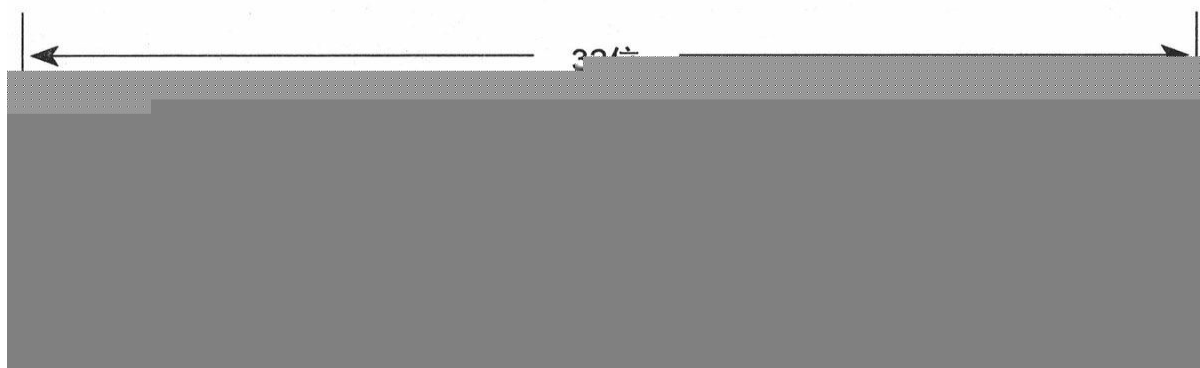


注意示例6-18和示例6-19中显示的路由器Laguna和Sandia的路由表，读者可以看出每一台路由器都学到了属于对应的RIPng进程的路由。路由器Taos不会把smallMountain进程的地址转发到bigMountain进程中，反之亦然。

示例6-18 IPv6路由表显示了路由器Laguna从路由器Taos的RIPng进程bigMountain学习到的路由



示例6-19 IPv6路由表显示了路由器Sandia从路由器Taos的RIPng进程smallMountain学习到的路由

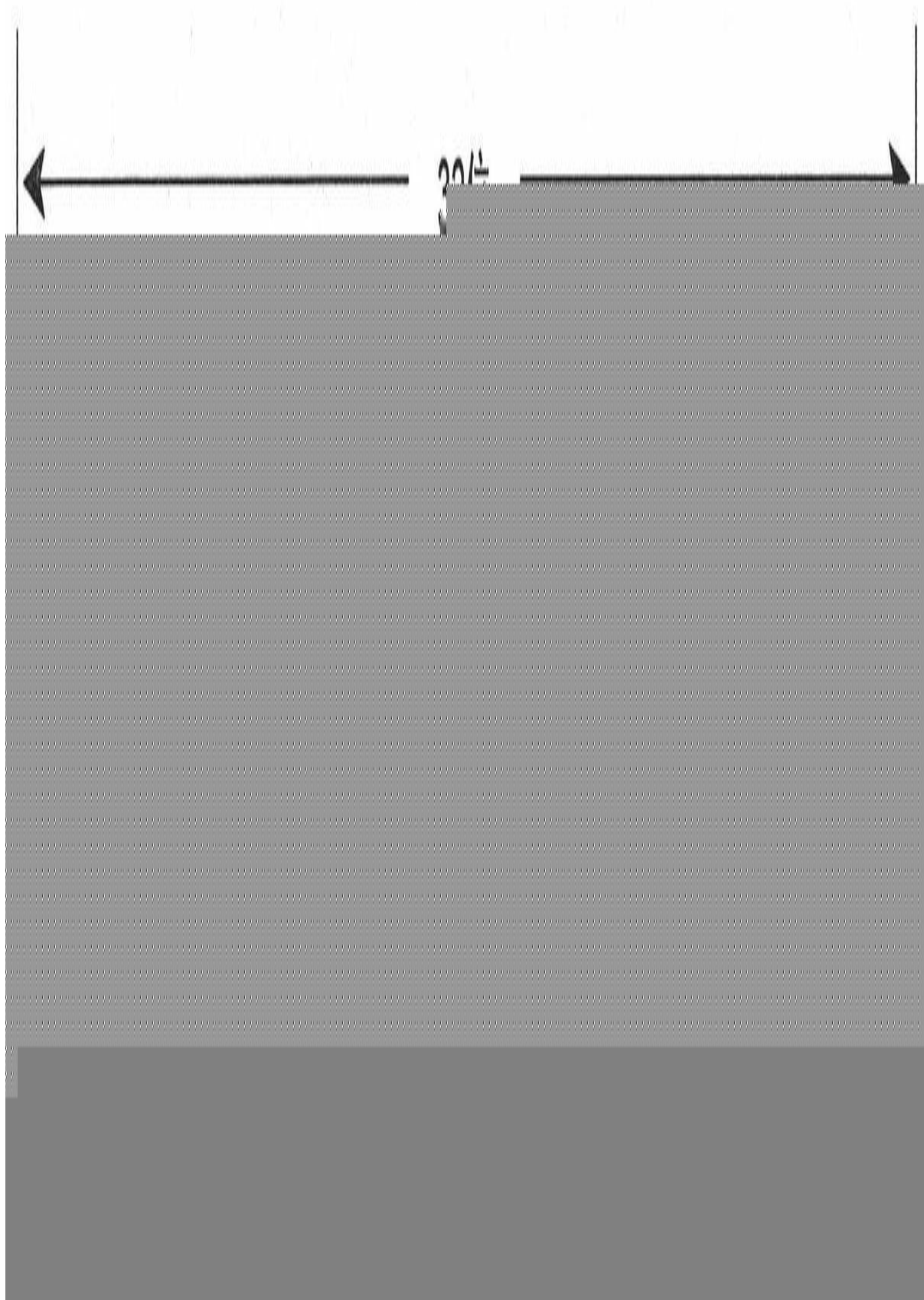


示例6-17中显示了路由器Taos学到了来自路由器Sandia的以太网接口Ethernet1上的4个地址，这些地址属于RIPng进程smallMountain:2001:db8:0:10::/64、2001:db8:0:11::/64、2001:db8:0:12::/64和2001:db8:0:13::/64。连接在接口Ethernet2上的地址2001:db8:0:5::/64将同时被通告到smallMountain和bigMountain进程中。在示例6-18中，路由器Laguna的路由表显示出没有加载任何smallMountain进程的路由。

参见示例6-20，在路由器Taos上打开RIPng的调试信息可以看到每一个接口上发送和接收的每一个RIPng进程的更新消息

示例6-20 命令**debug ipv6 rip**显示了每一个接口上发送和接收的每一个RIPng进程的更新消息





smallMountain进程的更新消息通过Ethernet1接口发送到路由器Sandia，通过Ethernet2接口发送到路由器Acoma。bigMountain进程的更新消息通过Ethernet0接口发送到路由器Laguna，通过Ethernet2接口发送到路由器Acoma。在Ethernet2接口上可以同时收到来自路由器Acoma的两个进程的更新消息。一个使用了缺省的UDP端口521，另一个使用了UDP端口527，后者是与smallMountain进程相关联的。使用527端口的更新包含了前缀2001:DB8:0:20::/64。这个前缀是分配给路由器Acoma的Ethernet1接口的，它是可以被路由器Sandia访问的。在Ethernet2接口上使用UDP端口521收到的更新消息包含了前缀2001:DB8:0:21::/64。这个前缀是分配给路由器Acoma的Ethernet2接口的，它可以分发到路由器Taos的bigMountain进程中，从而发送到路由器Laguna。

6.4.2 案例研究：RIPng进程的定制

路由选择进程的一些定制和RIPv1与RIPv2中的定制类似。读者可以使用全局配置命令**ipv6 router rip process_name** 配置RIPng进程的全局参数。

管理距离、路由选择进程用于负载均分的最大路径数，以及RIP计时器都已经在第5章中讲述了。这些参数在RIPng中的使用方法和在RIPv1与RIPv2中的用法是相同的。RIPng的缺省管理距离是120，这和RIPv1与RIPv2是一样的。RIPng用于负载均分的最大路径数的缺省值是16。RIPng能够负载均分的路径数最大为64。虽然并不是所有的计时器参数的值和RIPv1与RIPv2相同，但计时器参数是相同的：每30s更新一次，180s超时，保持计时器为0，垃圾收集计时器为120s。

参见示例6-21，路由器Taos的配置更改了管理距离、最大路径数和计时器。

示例6-21 路由器**Taos**的配置更改了管理距离、用于负载均分的最大路径数和协议计时器



示例6-22显示了命令**show ipv6 rip**的输出，显示了新的参数值。

示例6-22 在路由器Taos上，命令show ipv6 rip显示了更改后的RIPng参数值



比较bigMountain进程和smallMountain进程中的参数值，看看哪些仍然使用缺省值。在更改计时器和管理距离时应该特别小心。所有运行同一个RIPng进程的路由器都必须使用同样的计时器值。管理距离虽然只是在本地图由器上具有意义，但是如果路由器上正在运行多个路由选择协议或路由选择进程，并且地址是从多个路由选择进程学习到的，那么改变管理距离需要慎重考虑。更改路由选择进程的管理距离会改变它们的优先权。通过具有较低的管理距离的路由选择进程学习到的地址才会加入到路由表中。如果从具有较高的管理距离的进程学习到同样的地址，那么只有在第一个路由条目失效时才会将这个地址加入到路由表中。示例6-23中的配置将这些参数值改回了原来的缺省值。

示例6-23 在下面的配置中，路由器Taos的RIPng参数被恢复为原来的缺省值

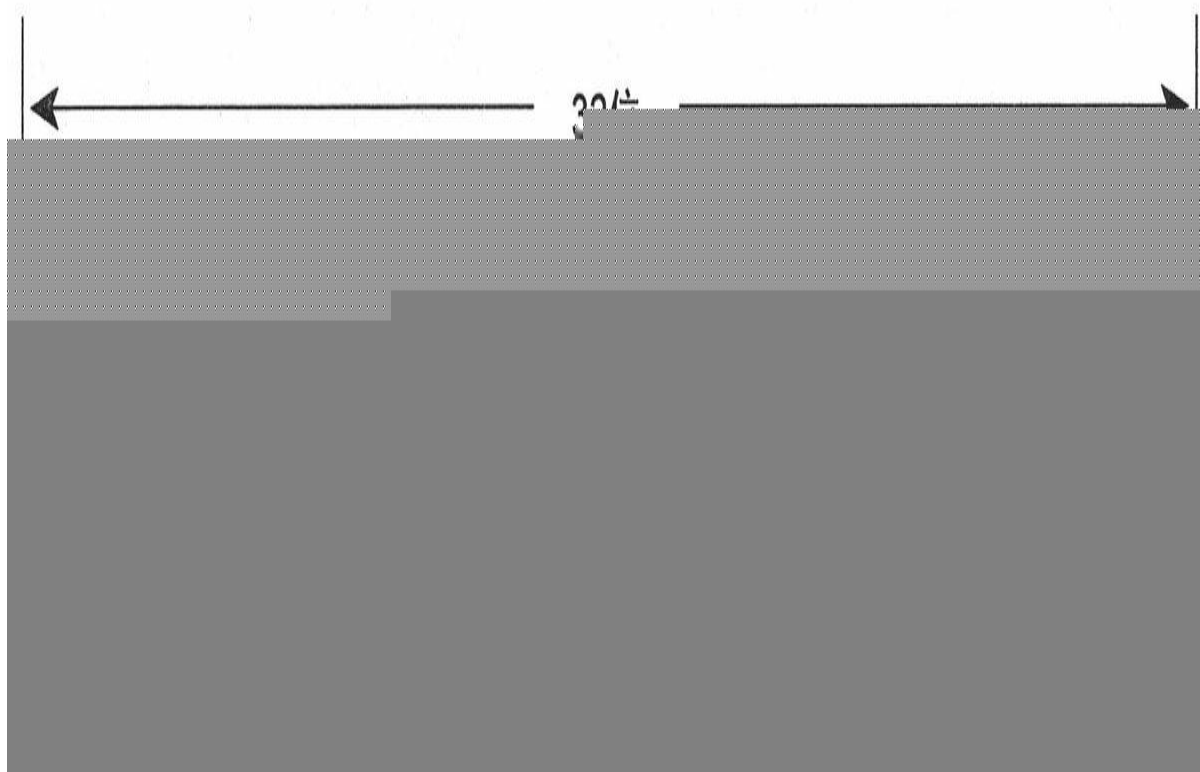


6.4.3 案例研究：RIPng的度量控制

RIPng与RIPv1和RIPv2一样，通过增加一个偏移量来调整它的度量值。但是，RIPng不是为前缀列表中的每一项路由条目增加度量，而是通过更改相关联的接口的跳数来实现的。也就是说，RIPng是通过在路由器相对应的接口上配置希望增加的跳数值来增加通过这个接口所通告的每一个前缀的度量值的。在缺省的情况下，RIPng为邻居路由器通告过来的前缀度量值增加一跳。在示例6-24中，显示了路由器Taos上的RIPng路由表和从路由器Sandia收到的一个RIPng更新消息。这里请注意，路由表中的每个条目的度量值都是2。路由表中的每个RIPng路由都是直接连接到距离只有一跳的路由器的，不论是连到路由器Sandia的还是Laguna的。路由器Sandia和Laguna是使用度量值1来通告它们的前缀的。路由器Taos为每一个接收到的路由度量值增加一跳。为了在路由器Taos上修改要添加到所收到的度量值，可以使用命令metric-offset来实现，参加示例6-25。

示例6-24 RIPng的路由表，使用调试命令**debug ipv6 rip**显示所收到的更新消息演示了接收到的度量上增加的度量偏移量



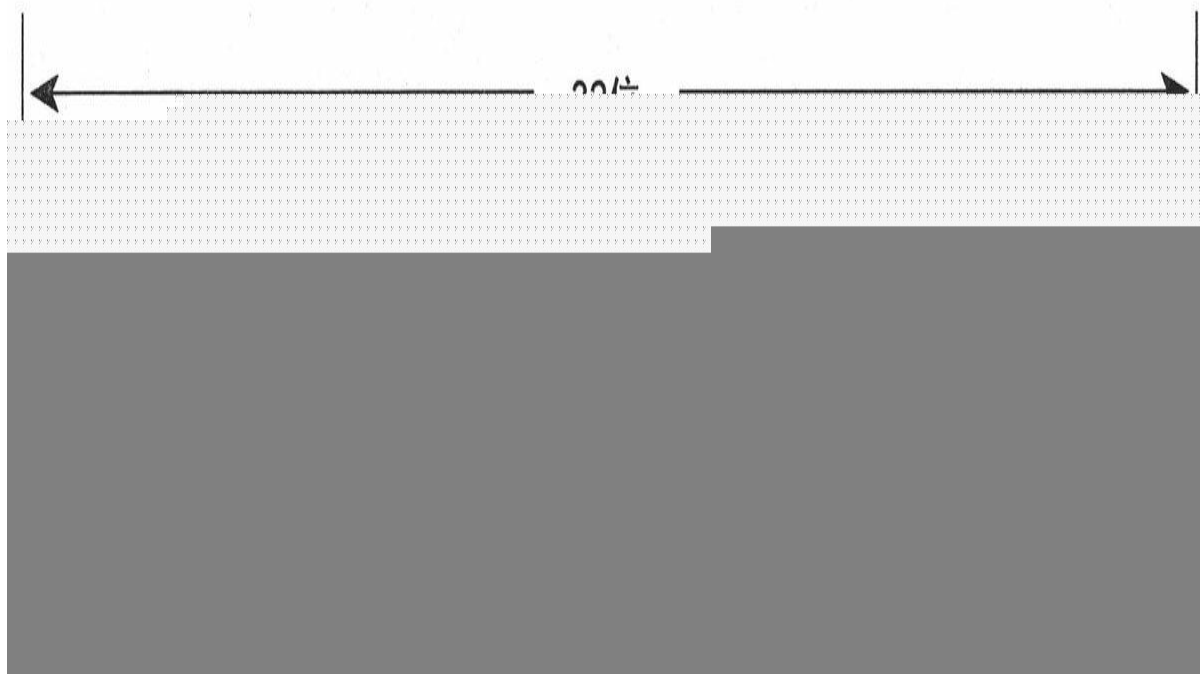


示例6-25 在路由器Taos的Ethernetl接口上将RIPng的度量偏移量增加为3



在示例6-26中显示了路由器Taos的新路由表。路由器Taos在Ethernetl接口上为它所收到的每一个前缀都增加了3跳。

示例6-26 在接收接口上更改度量偏移量后，RIPng路由表显示通过这个修改度量值的接口学到的每一个前缀都具有较高的度量值



这条命令允许我们调整与给定链路相关联的跳数值。通过调整后的接口所接收的所有地址将具有调整后的度量值。与之相比，RIPv1和RIPv2允许我们将在接口收到的一个子网（或所有）地址加入到路由表之前增加偏移量，或者这些地址被从该接口通告出去之前可以增加 偏移量。

6.4.4 案例研究：路由汇总

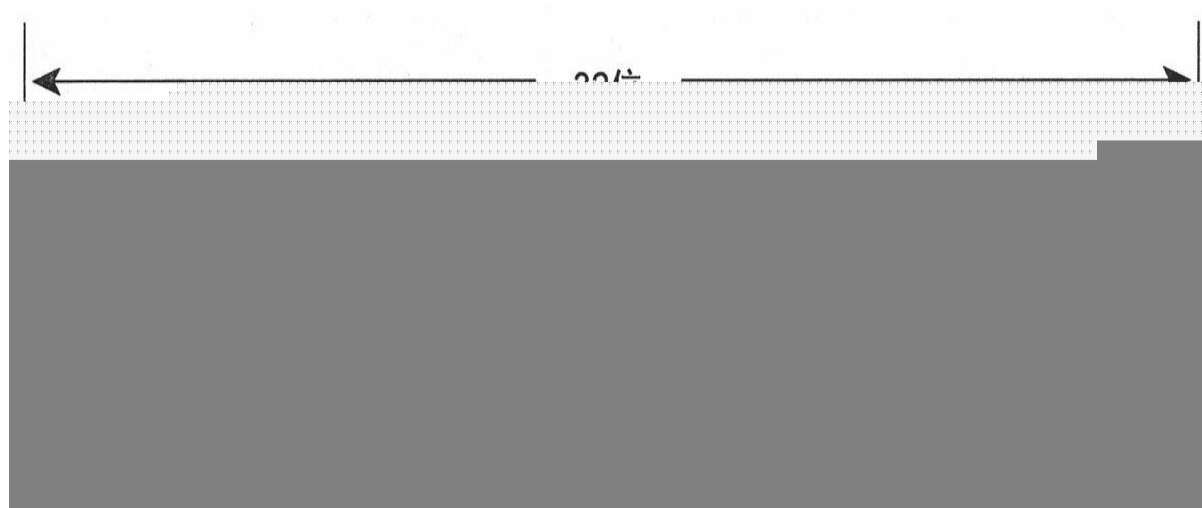
与RIPv2一样，RIPng的路由可以进行路由汇总后再通告给邻居。路由器Sandia需要通告地址2001:DB8:0:10::/64、2001:DB8:0:11::/64、2001:DB8:0:12::/64，以及2001:DB8:0:13::/64的路由。这些路由可以被汇总为一条路由：2001:DB8:0:10::/62。路由器Sandia的路由配置参见示例6-27。

示例6-27 路由器Sandia的配置对所通告的RIPng地址进行了汇总



在配置了覆盖这些具体路由前缀的汇总路由地址后，这些更具体的路由前缀将被自动地抑制。在示例6-28中显示了路由器Taos的新路由表。

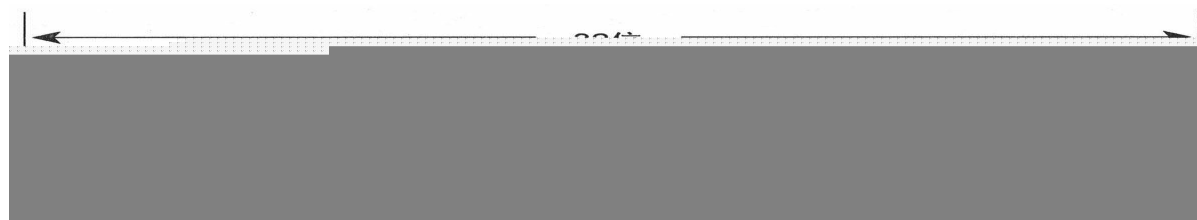
示例6-28 64位长的4个连续的前缀已经汇总为一条62位长的路由条目



6.5 RIPv2与RIPng的故障诊断

对于RIPv2协议，通常会碰到两个配置问题，即版本不匹配和认证配置错误。这两个问题都可以比较容易地从调试信息中发现，参见示例6-29所示。

示例**6-29** 通过调试信息来发现不匹配的版本和配置错误的认证问题



对于RIPv2或RIPng协议，或者任何无类别路由选择协议来说，更有可能出问题的原因是，配置了一个错误的可变长子网掩码。VLSM并不难，但是如果VLSM的规划没有小心地设计和管理，它就会带来一些非同寻常的路由选择困难。

案例研究：配置错误的VLSM

图6-17中的主机C不能通过网络进行通信，甚至无法在本地数据链路上ping通其他的主机或路由器。而主机A和主机B的相互通信没有问题，能够与网络上的其他主机正常通信，但是它们都不能和主机C通信。这里，所有的主机都把172.19.35.1配置成自己的缺省网关地址。

如图6-18所示，当从主机A或主机B上试图去ping主机C时，发现第一个ping是成功的，但是后续的ping是失败的。显然，至少有一个ICMP的Echo请求包能够到达主机C，并且至少有一个Echo响应包可以返回给ping的源主机，这一事实说明，网络的故障与硬件或数据链路没有太大关系。

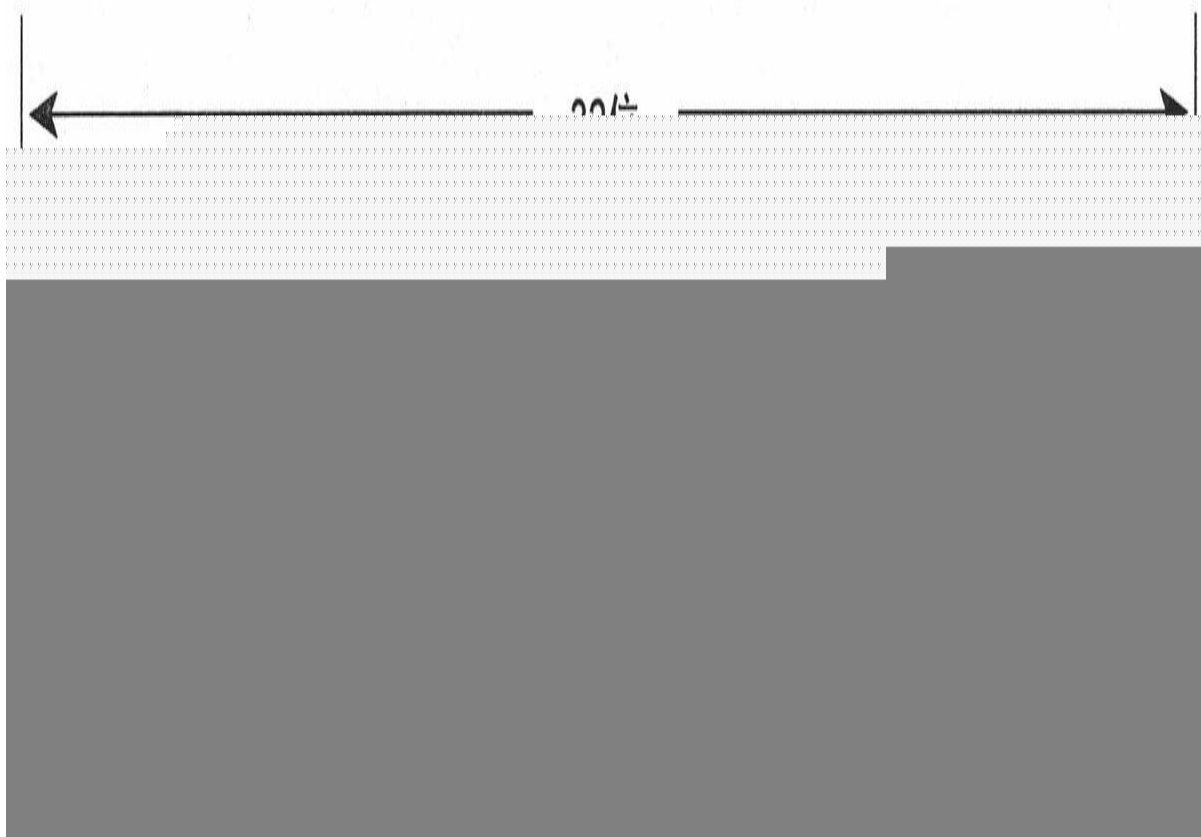


图6-17 主机A和主机B能够通过网络进行通信，但是都不能和主机C通信

这个ping的奇怪行为可以让我们作出这样一个假设：在第一个ping包成功后，后续的ping包——不论是主机B发出的Echo请求包，还是主机C返回的Echo响应包——不知由于什么原因被误导了。因为这种情况发生在本地的数据链路上，因此应该检查一下ARP（Address Resolution Protocol，地址解析协议）的缓冲区（cache）。

示例6-30和图6-19分别显示了主机C和主机B的ARP缓冲区。关于ARP的猜疑在这里得到证实，主机C的ARP缓冲区包含了主机B的正确MAC地址（00a0.2470.febd），但是主机B的缓冲区包含的与主机C的IP地址相关联的MAC地址是0000.0c0a.2aa9。进一步地 观察这两个缓冲区，显示出MAC地址0000.0c0a.2aa9是路由器San_Felipe的本地接口的MAC地址，从这个信息得知：通过路由器San_Felipe可以到达的目的IP地址和IP地址172.19.35.2映射到相同的MAC地址上了。

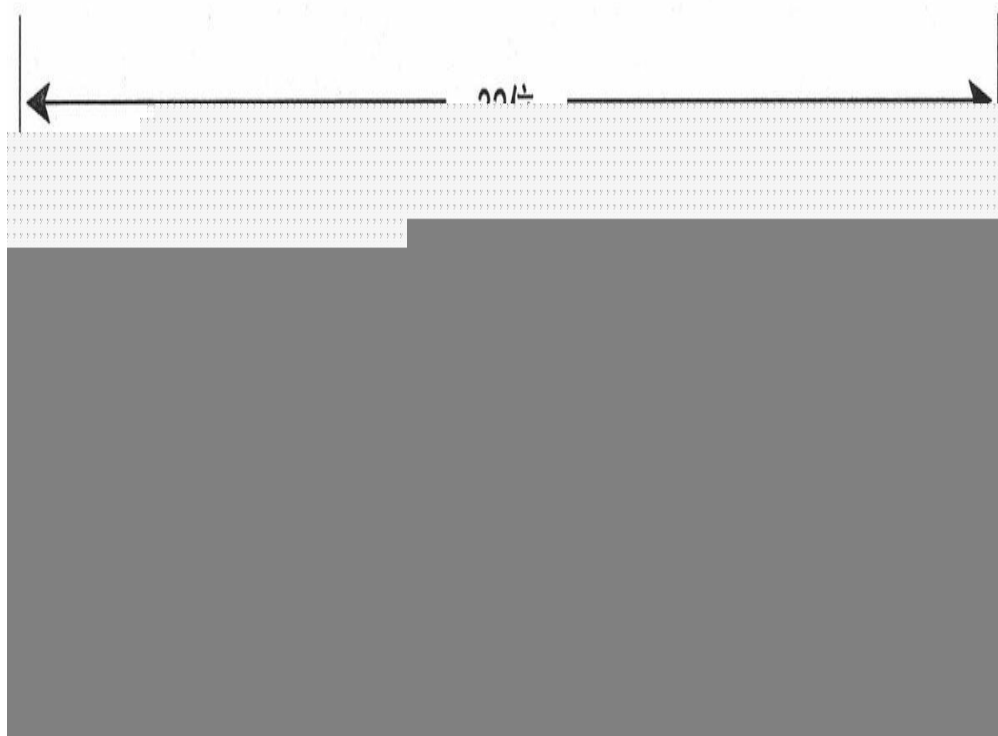
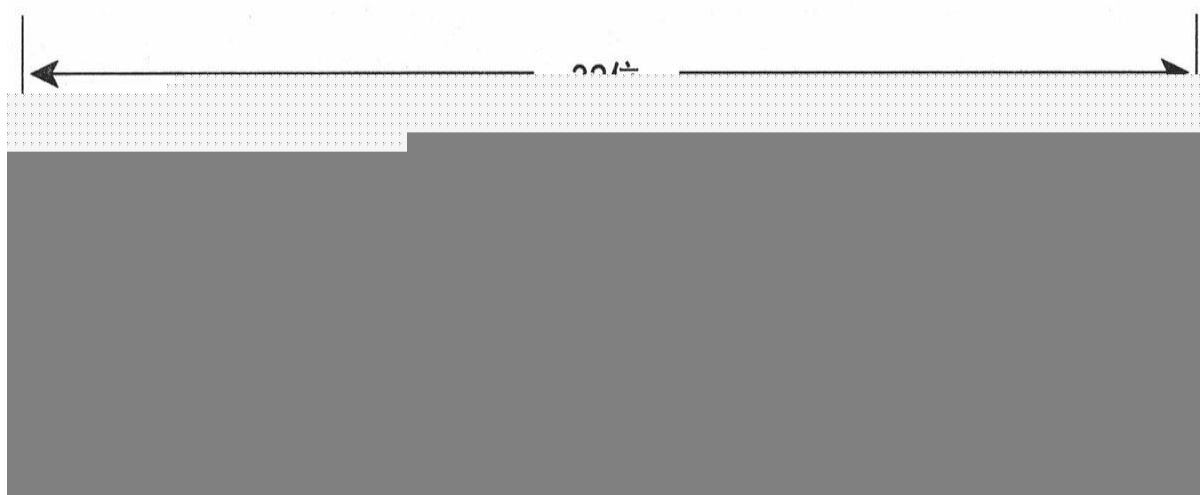


图6-18 当主机B试图ping主机C时，第一个ping包是成功的，而后续的ping包是失败的

示例6-30 主机C的ARP缓冲区正确地显示了所有地址的MAC地址



现在ping的结果就比较清楚了。首先，主机B广播了一个IP地址为172.19.35.72的ARP请求，然后主机C发送一个ARP响应包，因而主机B发送的第一个ping包是正确的。在这期间，路由器San_Felipe也收到了

那个ARP的请求包，很显然它认为自己有一条到达地址172.19.35.72的路由，于是路由器San_Felipe就用ARP代理（Proxy ARP）作出响应（滞后于主机C是因为路由器最初不得不执行一次路由的查找），这就导致主机B在自己的MAC缓存中覆盖了主机C的MAC地址。后来的Echo请求包被发送给路由器San_Felipe，而路由器San_Felipe将把这个请求包从本地链路路由出去，最终被丢弃；连接在这个以太网链路路上的协议分析仪证实了这一点（如图6-20所示）。

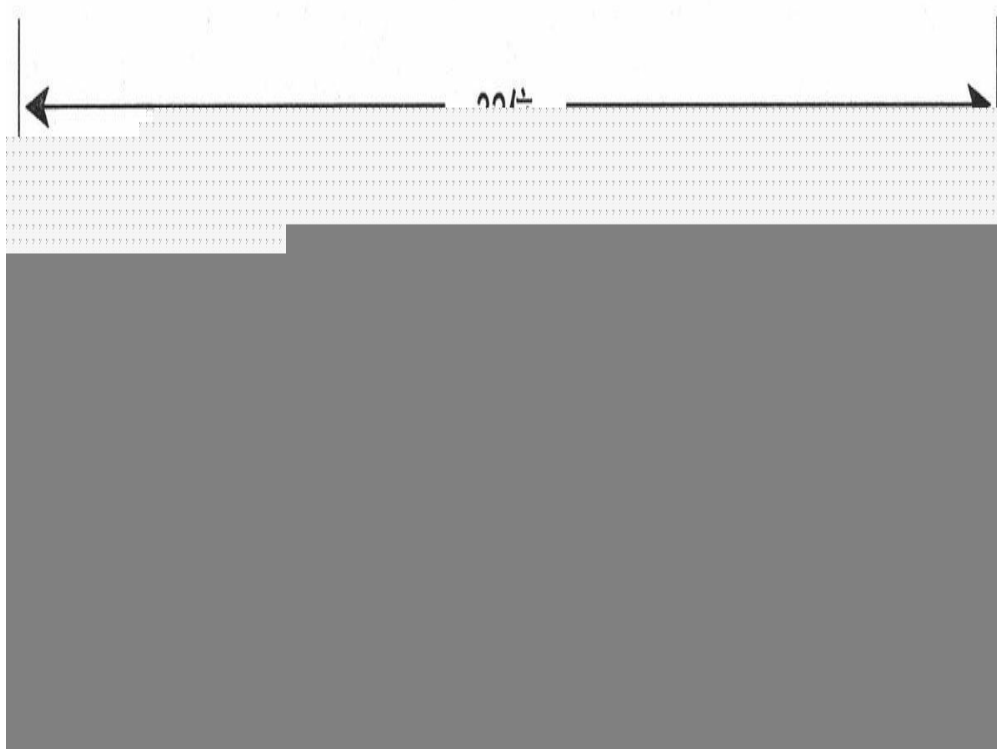


图6-19 主机B的ARP缓冲区显示，主机C的IPv4地址被映射到了路由器San_Felipe的本地接口172.19.35.2的MAC地址上

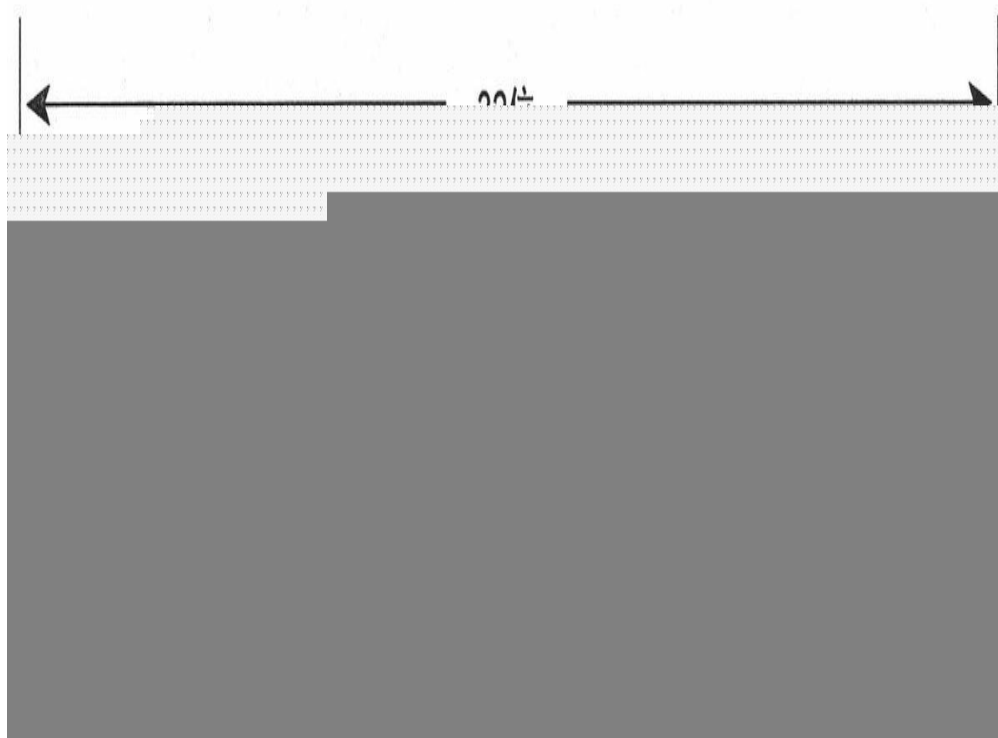


图6-20 协议分析仪过滤出的ARP数据包显示，主机B到主机C的ARP请求和从主机C（00a0.24a8.ala5）与路由器San_Felipe（0000.0c0a.2aa9）返回的响应

如果了解了故障是由路由选择问题引起的，那么余下的工作就是要找出引起路由选择问题的原因了。首先，应该确定一下每一条数据链路的子网地址，如图6-21所示。接着，基于二进制的表示，应该把主机C的IP地址和从路由器San_Felipe可达的所有子网相对照，来找出所有的地址冲突。在表6-5中，用粗体字显示了子网地址的最后一个八位组的子网位。



图6-21 当分析任何地址编址的规划时，特别是**VLSM**的设计，应该先确定每一条数据链路的子网地址，这样才能发现地址冲突和重叠的问题

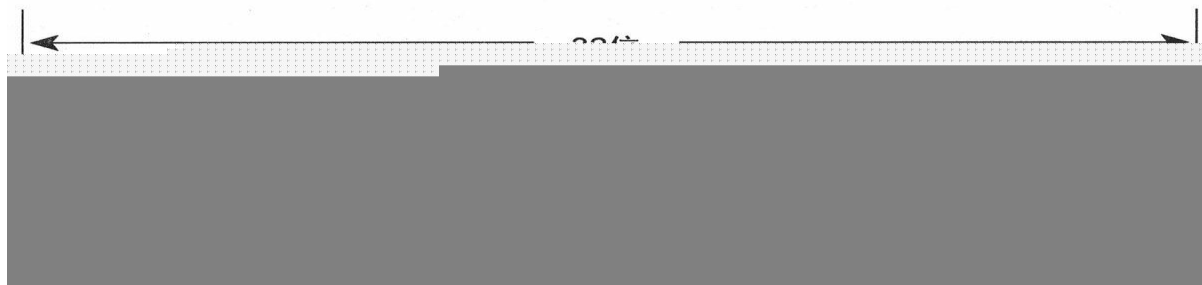
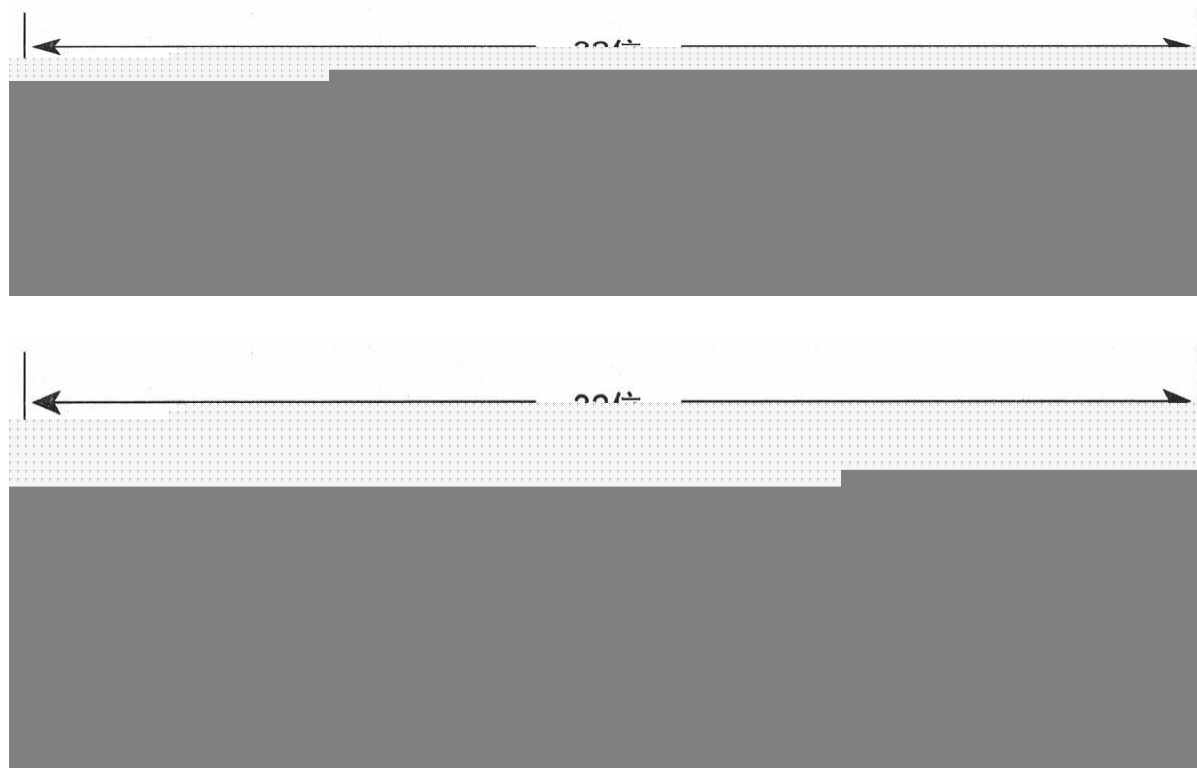


表6-5 路由器**C**的**IPv4**地址，高亮度处显示了最后一个八位组的子网位

经过比较后，显示出子网172.19.35.72/25的前3位和172.19.35.64/27的前3位是匹配的。路由器San_Felipe的路由表同时拥有172.19.35.0/25和172.19.35.64/27的路由（参见示例6-31）。当路由器收到一个要到达主机C的数据包时，它可以和子网172.19.35.0/25匹配1个比特位，但却可以和子网172.19.35.64/27匹配3个比特位。结果是，路由器将选择更具体的子网路由并把数据包从本地数据链路上路由出去，最后丢弃该数据包。

示例6-31 路由器**San_Felipe**同时拥有**172.19.35.0/25**和**172.19.35.64/27**的路由，第二个路由比第一个路由能够更好地匹配主机**C**的地址



该故障的解决办法是，要么更改主机C的地址，要么更改子网172.19.35.64。这一措施理论上听起来比较容易，但在实际的网络中，它会带来一些困难的决策，因为它涉及到这个案例研究中的网络上的其他客户。

表6-6中显示了网络172.19.35.0基于27位掩码的所有子网。这样做的目的是为了把最初的4个连续的子网捆绑成一个25位掩码的单个子网，以便容纳“骨干”以太网段上的85台主机。这种做法是可以的，因为地址编组所使用的所有子网的首个子网位都为0，因而没有其他的地址能够引起冲突；其次，子网172.19.35.192/27使用30位的掩码来划分更小的子网，给串行链路使用，这又一次证明该做法是有效的。子网172.19.35.128/27和172.19.35.160已经被使用了，当选择子网172.19.35.64/27和子网172.19.35.96/27给两个“远程”网段使用时，错误就产生了，因为这两个子网已经被宣告了。

这个困难的决策来自于，究竟是放弃骨干以太网段的地址空间进行重新编址，还是放弃那两个远程子网各自的地址空间进行重新编址？我们选

一个27位的掩码应用于子网172.19.35.0

6.6 展 望

虽然RIPv2协议比RIPv1协议有了很大的改进，但它依旧受到最大15跳的跳数限制，因此只能适用于小型网络。第7章、第8章和第10章将阐述在大型网络中使用的3种路由选择协议，并再结合像VLSM这样的设计策略，就可以成为控制大型网络的强有力的工具。

6.8 推荐读物

“RIP Version2”，即RFC 2453,Malkin.G于1998年11月编写。

“RIPng for IPv6”，即RFC 2080,Malkin.G和R.Minnear于1997年1月编写。

6.9 复习题

1. RIPv2的消息格式中包含了哪3个新的字段？
2. 除了复习题1中3个字段定义的扩展特性外，RIPv2相比RIPv1还有哪两个主要的改变？
3. RIPv2协议使用的多播地址是什么？多播方式通告消息与广播方式相比有什么好处？
4. 在RIPv2的消息中，路由标记字段的用途是什么？
5. 在RIPv2的消息中，下一跳字段的用途是什么？
6. RIPv2协议使用的UDP端口号是多少？
7. RIPv2协议使用的UDP端口号是多少？
8. 什么特性要求路由选择协议必须是一个无类别路由选择协议？
9. 什么特性要求路由选择协议必须使用VLSM？
10. 在Cisco的RIPv2中，可以使用哪两种类型的认证？它们都在RFC 2453中定义了吗？

6.10 配置练习

1. 在图6-12的例子中，路由器Taos配置成可以发送版本1和版本2的更新消息，因此Linux主机Pojoaque上的“routed”进程可以理解来自路由器Taos的更新。除了使用**ip rip send version** 命令，还有其他的方法配置路由器Taos吗？

2. 一个网络分配到地址192.168.100.0，划分这个地址来满足下面的需求：

- 1个含有50台主机的子网；
- 5个含有10台主机的子网；
- 1个含有25台主机的子网；
- 4个含有5台主机的子网。
- 10条串行链路

3. 配置图6-22中的4台路由器运行RIP协议。路由器RTC运行的版本是IOS 10.3，并由于策略原因不能升级。

4. 配置图6-22中的路由器RTB和RTD，使其在串行链路上对交换的RIP更新进行认证。

5. 配置图6-22中的路由器RTB和RTD，使其在配置练习4的认证钥匙生效后的3天改用一个新的认证钥匙，这个新钥匙生效10小时后再改用另一个钥匙。

6. 配置图6-22中路由器RTA和RTB运行RIPng。在接口上分配下面的IPv6前缀：

RTA:

2001:DB8:0:1::/64

2001:DB8:0:2::/64

RTB:

2001:DB8:0:3::/64

2001:DB8:0:2::/64

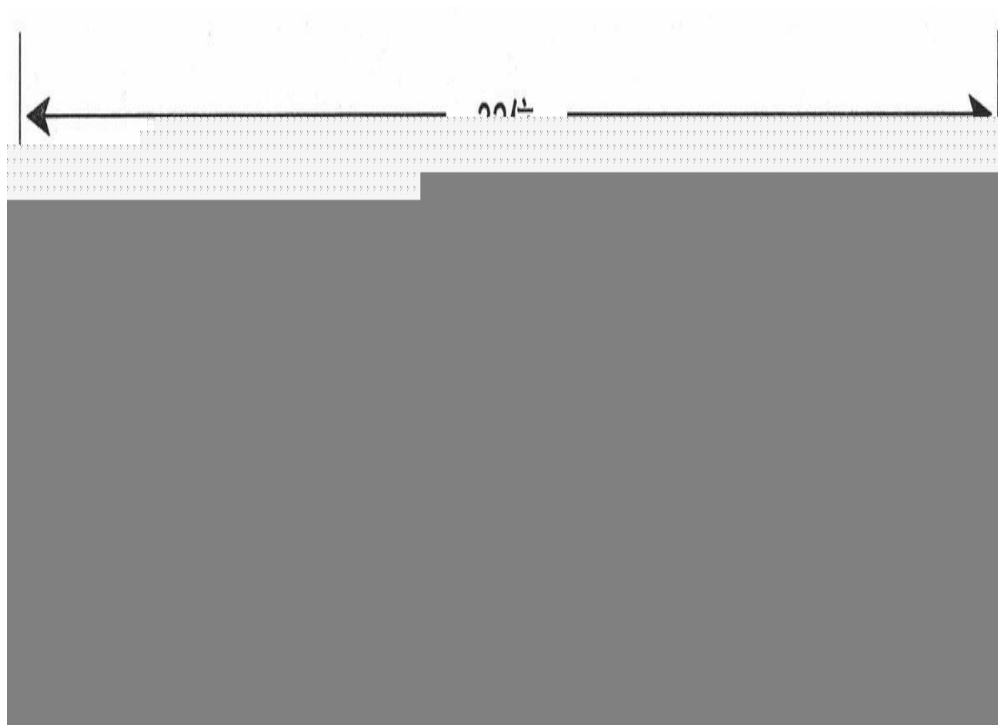


图6-22 配置练习3~6的网络

6.11 故障诊断练习

1. 示例6-32～示例6-34显示了图6-23中的3台路由器的配置。哪些子网出现在每一台路由器的路由表中？在每一台路由器上，哪些子网是可达的？哪些子网（如果有的话）是不可达的？
2. 将图6-23中的路由器RTA和RTB的配置改变如下： [\[14\]](#)

```
interface Ethernet0
```

```
ip address 192.168.13.35 255.255.255.224
```

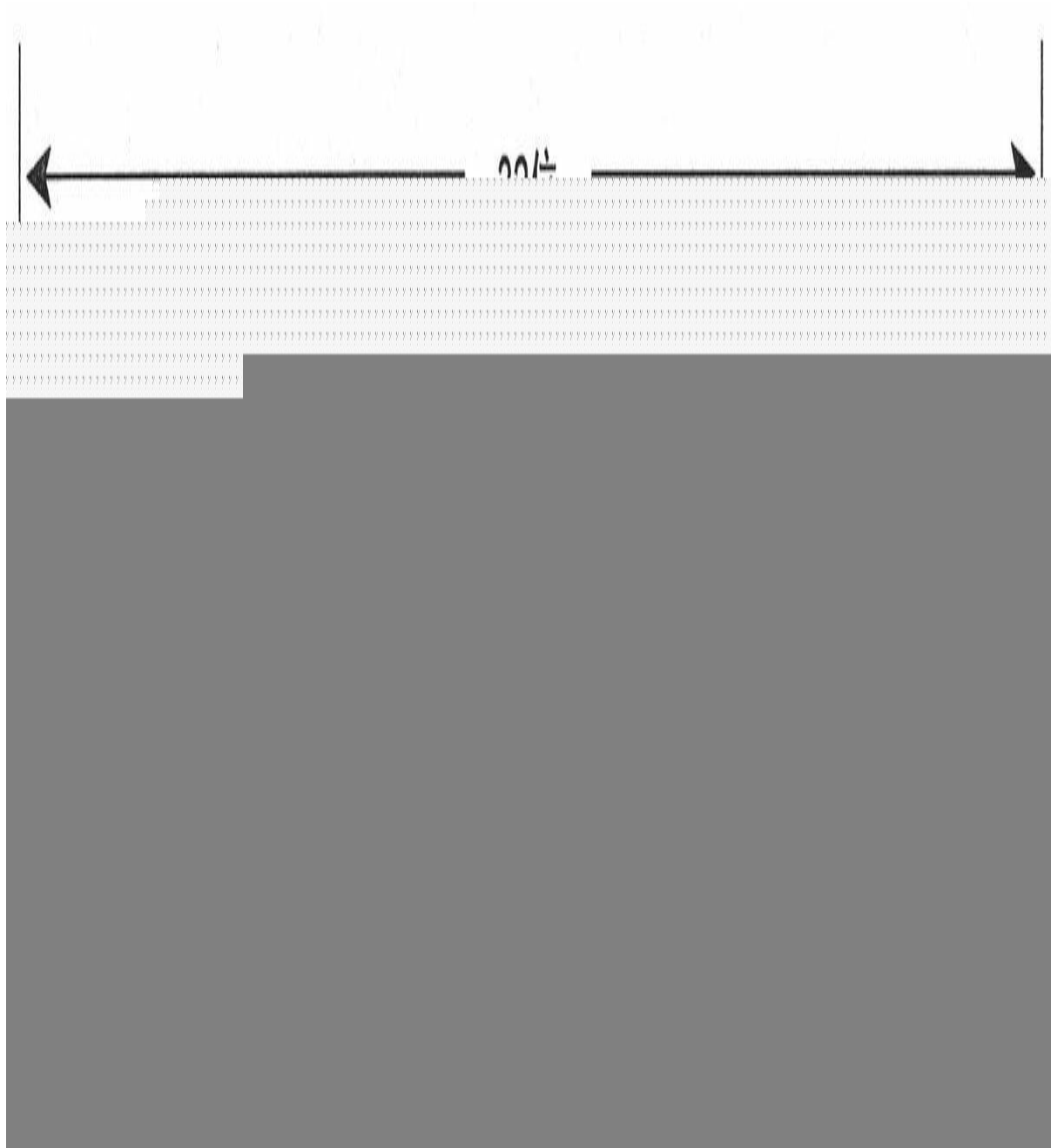
```
ip rip receive version 1 2
```

这个改变配置的结果会有一些子网增加到路由表中吗？解释一下为什么有或为什么没有？

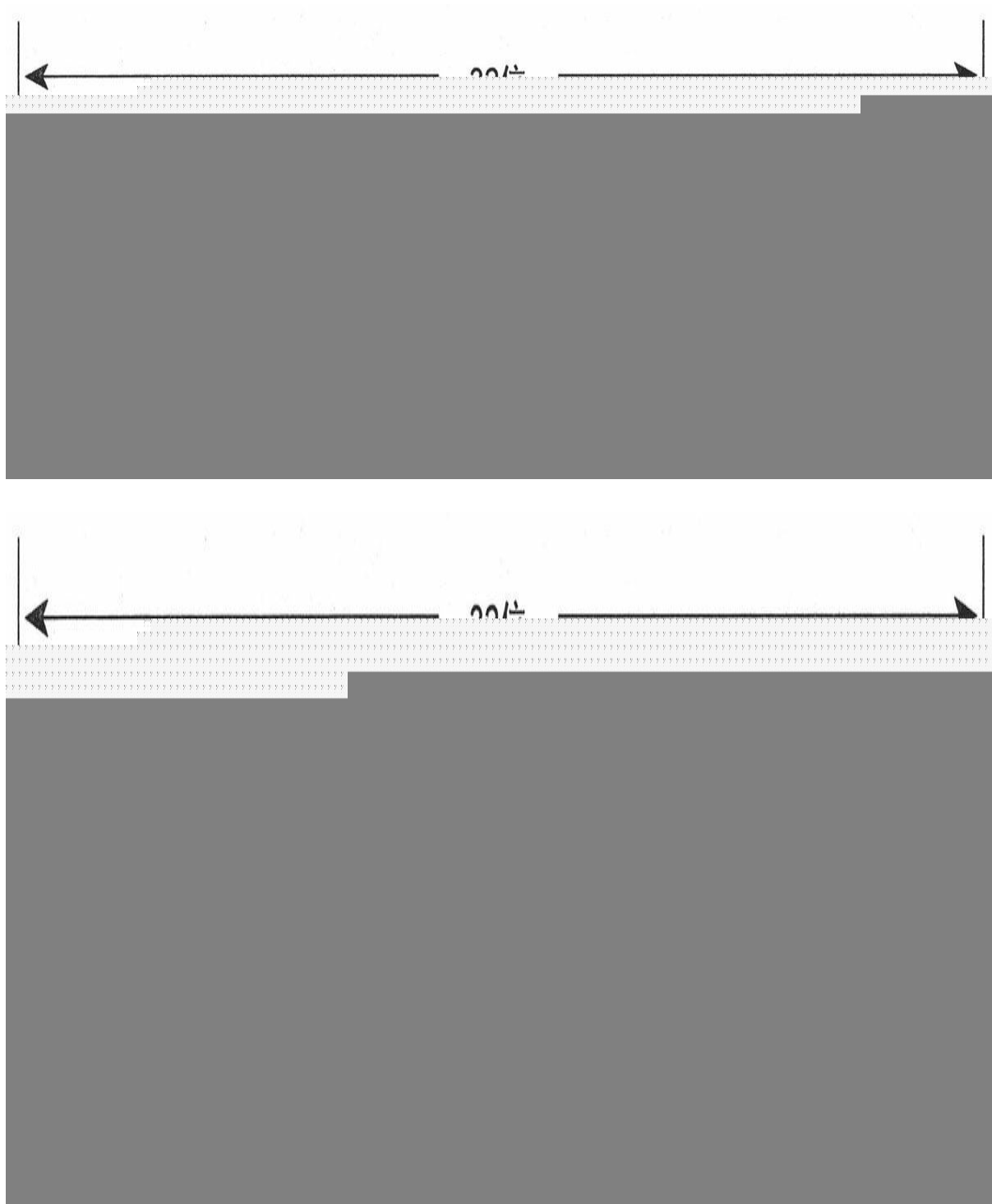


图6-23 故障诊断练习1～2的网络

示例6-32 图6-23中路由器RTA的配置



示例6-33 图6-23中路由器RTB的配置



示例6-34 图6-23中路由器RTC的配置



[1] 对这个RFC的补充还有RFC1721——“RIP Version 2 Protocol Analysis”和RFC1722——“RIP Version 2 Protocol Applicability Statement”。

[2] 该图是根据RFC 1722中Gary Malkin演示的一个例子改编的。

[3] 重新分配是指把从一个路由选择协议学习到的路由通告给另一个路由选择协议的情况，这将在第11章中详细讨论。

[4] 这种方法只适用于下面的情形，即路由器上没有其他运行RIP协议的接口和相同的主网相连。

[5] 这个说法假定全0和全1的子网是可以被路由的，因为对于只有1位被子网化时，全0和全1子网是惟一可用的子网。

[6] 强烈建议读者把这个例子的所有地址转换成二进制的。

[7] 子接口超出了本书的讲解范围，还不熟悉这些有用工具的读者可以参考Cisco配置手册。

[8] MD5是在RFC1321中描述的。欲更好地了解MD5，请参阅下面这本书：*Network Security: Private Communication in a Public World* 的第120~122页，由Charlie Kaufman、Radia Perlman和Mike Spencer撰写，1995年Prentice Hall出版。

[9] 实际上，这个例子中使用“routed-s”选项是故意配置错误的。

[10] 现在，子网的概念应该比较熟悉了，从这里开始，子网作为一个单纯的术语将用来表示一个子网、一个子网的子网、一个子网的子网的子网，等等。

[11] 将几个地址合并成一个地址的技巧将在第7章的地址聚合中介绍。

[12] NTP协议超出了本书的讲述范围，请参考Cisco配置手册以便获取更多的信息。

[13] MPLS VPN允许多个RIP进程运行在单个提供商的边缘路由器（Provider Edge Router, PE）上。每一个给定的VPN配置一个单一的RIP进程。MPLS和VPN的内容已经超出了本书的讲述范围。

[14] 显示的是路由器RTB的配置，除了IP地址192.168.13.34外，路由器RTA的配置和路由器B是相同的。

本章包括以下主题：

- EIGRP的前身：IGRP协议回顾；
- 从IGRP到EIGRP；
- EIGRP的基本原理与实现；
- 配置EIGRP；
- EIGRP故障诊断；

第7章

增强型内部网关路由选择协议（EIGRP）

增强型内部网关路由选择协议（Enhanced Interior Gateway Routing protocol, EIGRP）是在Cisco IOS 9.21版中首次发布的，顾名思义，它是Cisco内部网关路由选择协议（Cisco Interior Gateway Routing protocol, IGRP）的增强版。这个命名是比较恰当的，因为它不像RIPv2协议那样，EIGRP协议对IGRP协议所增加的扩展特性远远多于RIPv2对RIPv1的扩展。和IGRP协议一样，EIGRP协议依然是一个距离矢量协议，并且使用了IGRP协议所用的复合度量。除此之外，EIGRP协议和IGRP协议几乎没有更多的相似之处。

IGRP协议在IOS软件系统的12.2（13）T和12.2（Rls4）S版本中已经停止使用。虽然是技术创新不断的时代，那些希望比RIP能够提供更多性能的现代网络操作人员已经转移到EIGRP协议或OSPF协议，而不是转移到改进一点性能的IGRP协议。但是，在学习EIGRP协议之前，首先了解一下该协议的发展还是有用的。

7.1 EIGRP的前身：IGRP协议回顾

20世纪80年代中期，作为对RIP协议局限性的回应，Cisco公司开发了IGRP协议，其中最重要的变化就是跳数的度量和15跳对网络口径大小的限制。IGRP通过多种路由变量参数计算出一个复合型的度量，并提供一些“旋钮”供这些变量参数施以权重，以便反映和衡量网络的一些特征和需求。虽然跳数并不作为这些变量参数之一，但IGRP协议还是延续使用了跳数，并且能够实现网络最大为255跳的需求。

IGRP协议相对于RIP协议还有其他一些优点：

- 非等价负载均衡（unequal-cost load balancing）；
- 更新周期是RIP协议的3倍时间长；
- 更新数据包格式更有效。

IGRP协议和EIGRP协议的一个主要缺点是它们是Cisco公司的私有协议，因此，只能在Cisco公司的产品平台上使用，而RIP协议则可以作为所有平台上的任何IP路由选择进程的一部分来使用。

Cisco公司开发IGRP协议的主要目的是创建一个功能强大的通用协议，以便使它能适应已选路由协议簇的多样性。IGRP协议除了作为IP路由选择协议，它还适用于ISO无连接网络协议（Connectionless Network Protocol, CLNP）的路由。这个适应于多协议的性能也是EIGRP协议的特性，它不仅可以用来进行IP网络的路由，也可以用于IPX和AppleTalk网络的路由。

从宏观的角度来看，IGRP协议继承了许多RIP协议的操作特点。IGRP协议也是一个有类别距离矢量型协议，除了被水平分隔法则抑制的路由外，IGRP将不断地周期性地向邻居路由器广播它的整张路由选择表。像RIP协议一样，路由器启动时，IGRP在所有运行IGRP协议的接口上广播出一个请求数据包，并对收到的更新执行一个完整性的检查，用来验证更新数据包的源地址是否和收到更新的那个子网属于同一个子网。^[1]新的带有可达路由度量值的路由更新条目将会被放置在路由表中，并且仅当它所带的度量值小于到达相同目的地址的原有路由条目的度量值

时，才能替代原有的路由条目。IGRP协议同样使用带毒性反转的水平分隔法则、触发更新和抑制计时器等手段来保证它的稳定性；同RIP协议一样，IGRP协议也在网络边界上进行地址汇总。

与RIP协议不同，IGRP协议不使用UDP来访问数据包，它直接通过IP层的协议号9来进行数据包访问。

7.1.1 进程域

IGRP协议也使用进程域的概念。通过定义和跟踪多个进程域，我们可以把一个域内的通信和另一个域内的通信隔离开来。域间的流量可以通过路由重新分配（第11章）和路由过滤（第13章）来严密地控制。

图7-1显示了进程域和路由选择域的对照。在这里定义了两个自主系统（AS）：AS 10和AS 40，这些系统是路由选择域——在一个共同的管理机构下运行的一个或多个IGP协议的路由器集合。它们通过外部网关协议（Exterior Gateway Protocol，在这个实例中，是边界网关协议，或称为BGP协议）来通信。

在自主系统AS 10中有两个IGRP进程域：IGRP 20和IGRP 30。在IGRP协议内，定义了两个自主系统号20和30，就此处而言，这些数字是用来区分同一个路由选择域内的两个不同路由选择进程的。进程域IGRP 20和进程域IGRP 30是通过和这两个进程域都相连的一台路由器来进行通信的。这台路由器同时运行两个IGRP进程，并且在这两个进程之间自动地进行路由再分配。

在IGRP的路由更新消息中，IGRP把路由条目分成3类：内部路由（interiorroute）、系统路由（system route）和外部路由（exterior route），每个IGRP的路由条目都属于这3个类别中的一个。

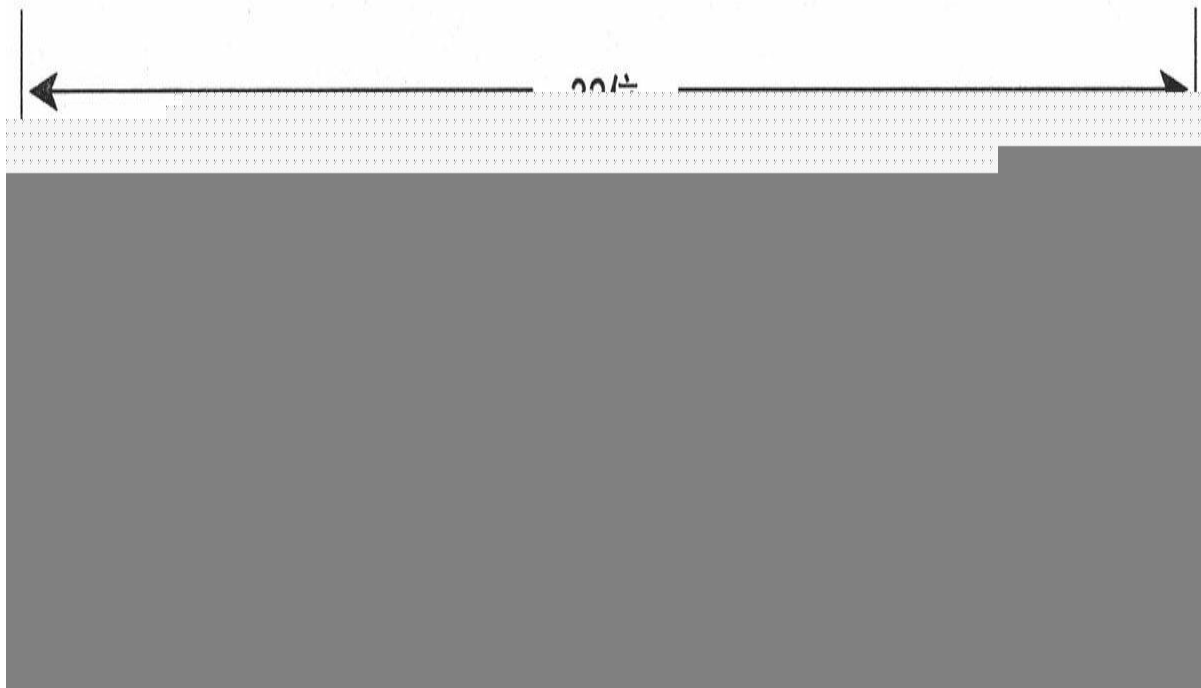


图7-1 一个自主系统号可以指定一个路由选择域，即在一个单一的管理域内运行一个或多个**IGP**协议的一组路由器集合。在**IGRP**中，一个自主系统号也可以指定一个进程域，即依赖于一个单一的路由选择进程共享路由选择信息的一组路由器

- **内部路由（Interior Route）** ——是指到达属于某个主网络的子网地址的路径，这里的主网络是指正在广播这条路由更新的数据链路的主网络地址。换句话说，作为内部路由被通告的子网对于通告路由器和接收路由器共同相连的主网络来说是“本地”的。
- **系统路由（System Route）** ——是指到达在网络边界路由器上被汇总的网络地址的路径。
- **外部路由（Exterior Route）** ——是指到达被标记成缺省网络（default network）的路径。对于缺省网络，路由器将直接发送所有的数据包而不对更具体的目的网络进行查找匹配。[\[2\]](#) 缺省网络和其配置方法将在第12章中讲述。

图7-2显示了IGRP协议是怎样使用这3类路由的。路由器LeHand和Tully都与子网192.168.2.64/26相连，所以主网络192.168.2.0被认为是这两台路由器“共享”的“本地”网络。而路由器LeHand和子网192.168.2.192/26相

连，显然子网192.168.2.192/26是连接路由器LeHand和Tully的主网络192.168.2.0的另一个子网。因此，路由器LeHand把子网192.168.2.192/26作为内部路由通告给路由器Tully。

然而，与路由器LeHand和路由器Thompson相连的本地网络却是192.168.3.0。由于路由器LeHand是主网络192.168.2.0和192.168.3.0的边界路由器，因此网络192.168.2.0被作为一条系统路由通告给路由器Thompson，同样的，网络192.168.3.0也被作为一条系统路由通告给路由器Tully。

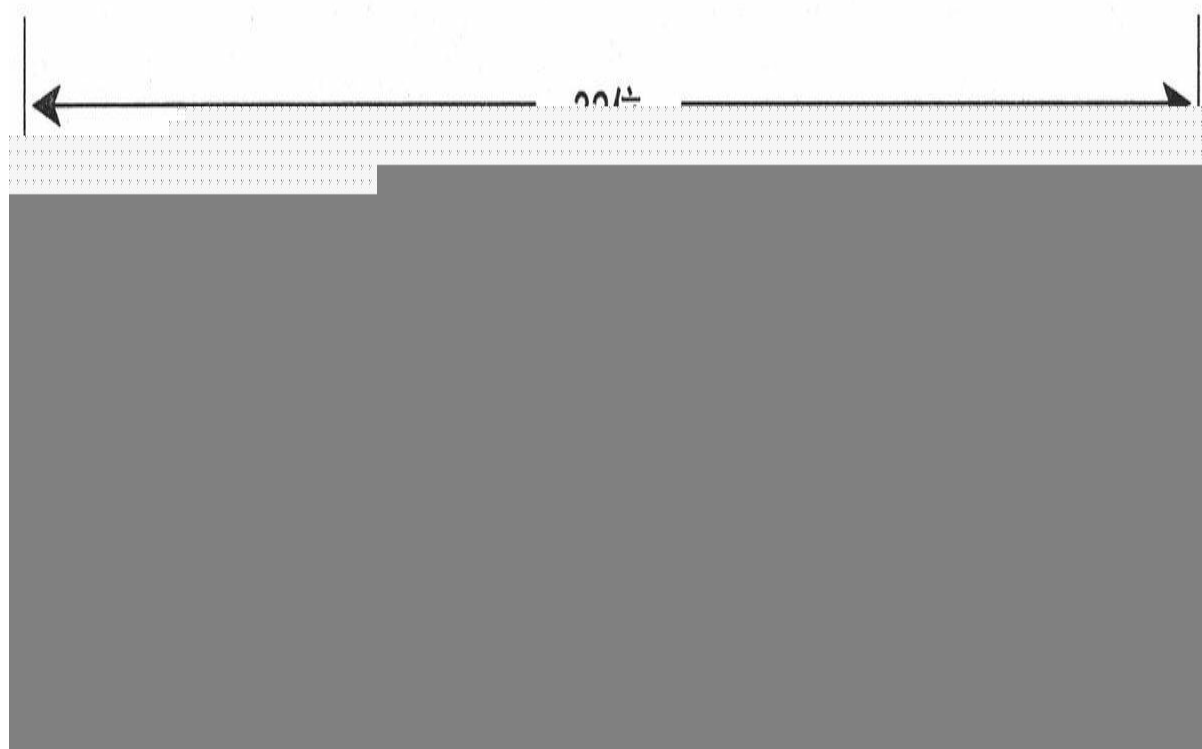


图7-2 路由器LeHand把子网192.168.2.192/26作为一条内部路由通告给路由器Tully，把网络192.168.3.0作为一条系统路由通告给路由器Tully，并把网络192.168.1.0作为一条外部路由通告给路由器Tully

网络192.168.1.0属于另一个自主系统，因而路由器LeHand把它作为一条缺省网络地址来通告，而192.168.1.0就被作为一条外部路由通告给路由器Tully和Thompson。

7.1.2 IGRP的计时器和稳定性

IGRP协议的更新周期是90s。为了防止更新计时器的同步，IGRP针对每一个更新时间都减掉一个最大为其20%的随机抖动变量；因此，每个更新周期所需要的时间将在72~90s之间变化。

当一条路由首次被用时，这条路由的无效计时器就会被设置成270s，即是更新周期的3倍时间。同时，刷新计时器设置成630s，即是更新周期时间的7倍长。每次接收到路由的更新报头后，这些计时器都将被重新初始化。如果在收到一条更新报头之前无效计时器的计时超时了，这条路由就会被标记成不可到达。但是，在路由器的刷新计时器超时前，这条路由还会被保留在路由选择表中，并且作为不可达的路由通告出去；如果刷新计时器超时了，这条路由才会从表中删除掉。

与RIP协议时长为30s的计时器相比，IGRP协议使用了90s的计时器，这意味着IGRP协议的周期性更新将比RIP协议占用更少的带宽。然而，这是在同样的情况下，IGRP协议比RIP协议的收敛更慢为代价的。例如，如果一台路由器离线了，IGRP协议需要3倍于RIP协议的时间才能检测到这个已不存在的邻居。

如果一条路由的目的地址变为不可达，或下一跳路由器增大了到达目的地址的度量以至于引起一个触发更新的话，那么这条路由将会进入一个280s（3倍的更新周期加上10s）的抑制时间状态。直到抑制计时器超时之前，有关这个目的地址新的信息都不会被路由器接受。IGRP协议的抑制特性可以用命令`no metric holddown`来禁止。在一个没有路由环路的网络拓扑中，抑制特性没有实际的意义，禁止该特性将有助于减少IGRP的收敛时间。

缺省的计时器可以用下面的命令来改变：

timers basic *update invalid holddown flush* [sleeptime]

除了 ***sleeptime*** 选项，这条命令曾在改变RIP协议的计时器时使用过。***Sleeptime*** 是一个周期性的毫秒（ms）级的计时器，在收到一条触发更新后，它被用来延迟一个正常的路由选择更新。

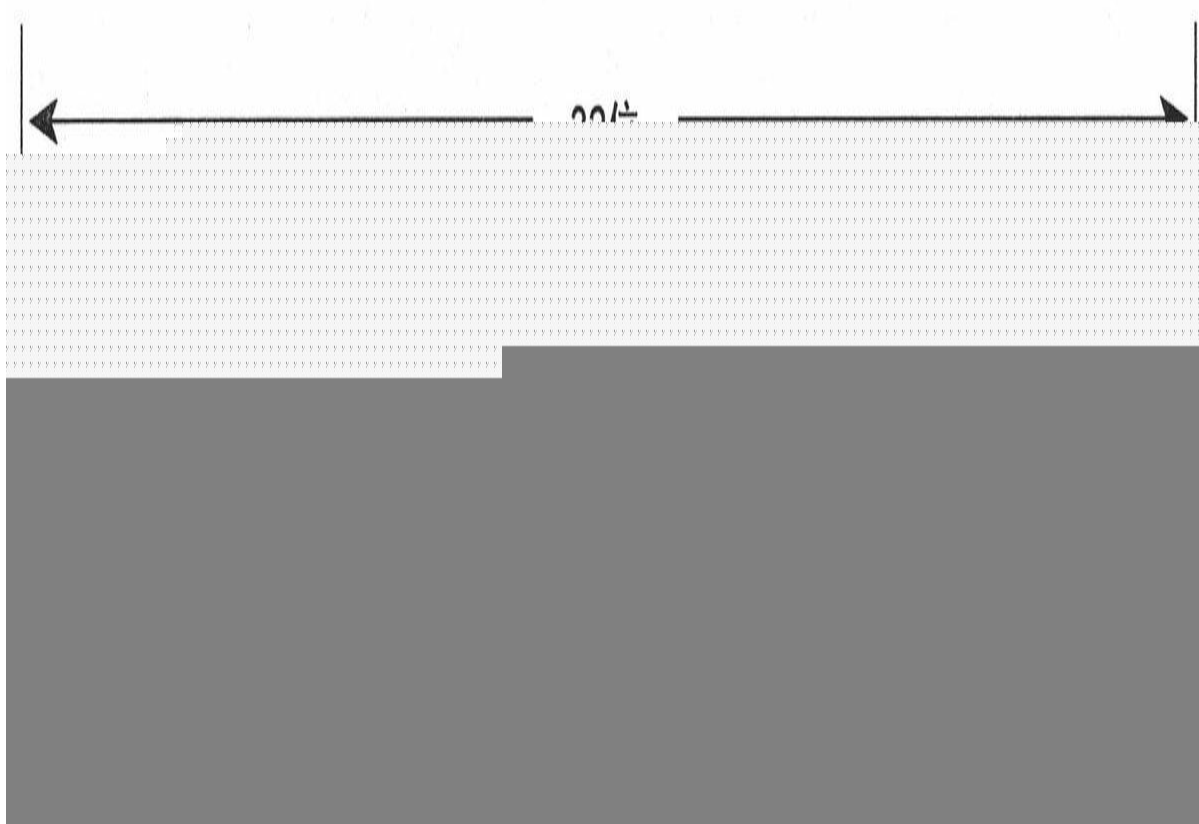
7.1.3 IGRP的度量

与RIP协议相比，IGRP协议引入的一个最重要的变化是它基于链路的特

征，使用了多种度量参数，这些也继承到了EIGRP协议中。学习IGRP协议如何运用这些度量参数，对于掌握EIGRP协议怎样运用它的度量参数是有用的。

在IGRP协议中，IGRP将根据链路的特性计算出一个“复合”的度量值，这些链路特性包括链路带宽、时延、负载和可靠性。缺省条件下，IGRP选用路由的链路带宽和时延作为度量值。如果把数据链路想象成一个管道，那么带宽就像是这个管道的宽度，而时延就像是这个管道的长度。换句话说，带宽是数据传送能力的量度，而时延是端-端传送时间的量度。链路的另外两个特性——负载和可靠性只有在路由器上进行人工配置后才会被应用。虽然IGRP协议不使用MTU（Maximum Transmission Unit，最大传输单元）作为计算复合度量值的参数，但IGRP也会跟踪每条路由上的最小MTU的大小。参见示例7-1，可以通过命令show interfaces来观察一个特定接口上有关IGRP的复合度量值的大小。

示例7-1 show interface命令的输出包含了关于这个接口的度量的统计。这个以太网接口的度量值显示MTU=1500字节，带宽=10Mbit/s，时延=1000μs，可靠性=100%，负载=39%（最小负载）



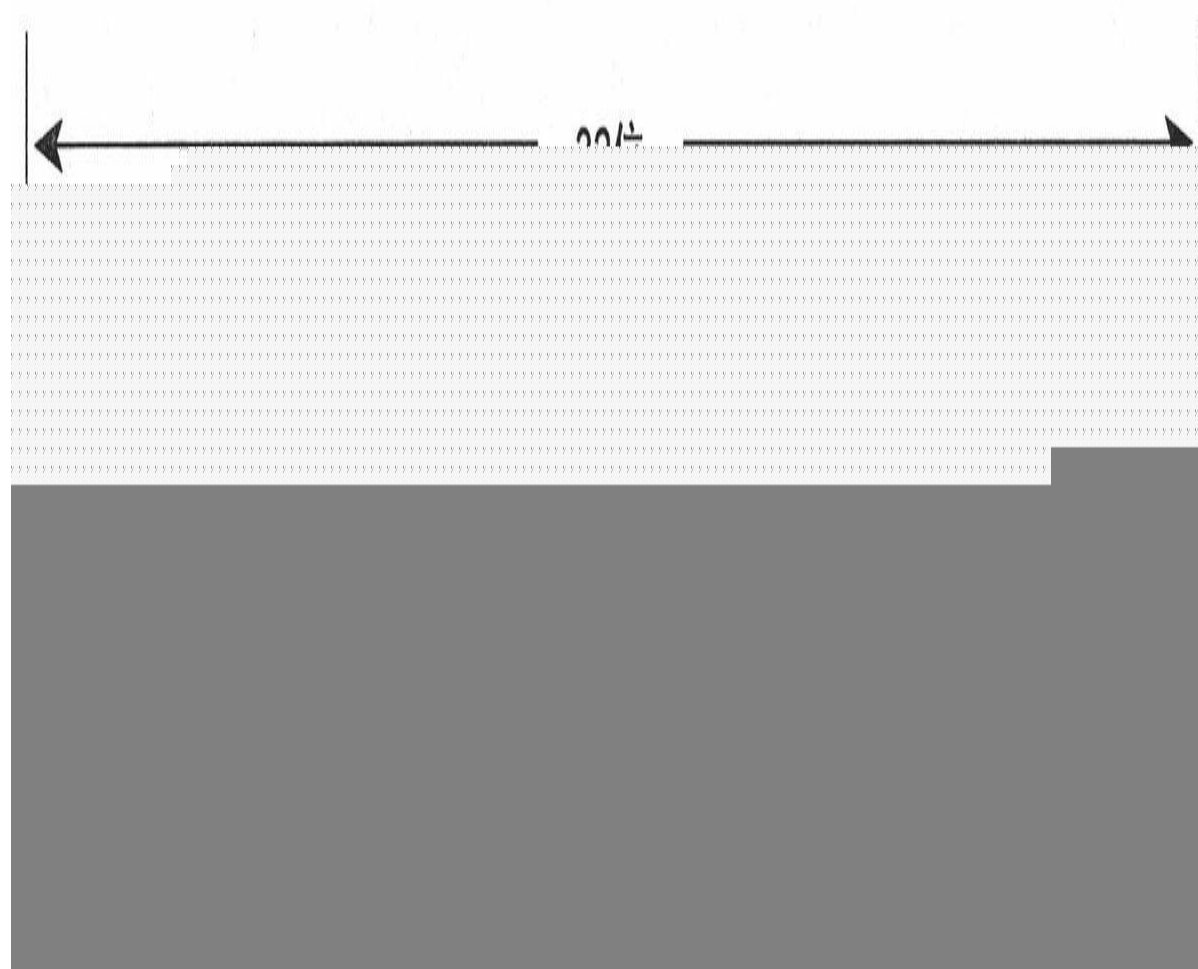
带宽用kbit/s单位来表示，它在计算链路的度量值时仅作为一个静态的值，没有必要反映出链路实际使用的带宽；也就是说，带宽不需要动态地去量度。例如，不论和串行接口相连的是T1的还是56KB的链路，串行接口的缺省带宽都是1 544kbit/s。这个缺省的带宽值可以通过bandwidth命令来更改。

IGRP的更新消息使用3个八位组字节来表示IGRP“带宽”，在本书中用BW_{IGRP}表示，它是用因子 10^7 除以带宽得来的，因此，如果接口的带宽是1544，那么

$BW_{IGRP} = 10^7 / 1\,544 = 6\,476$ ，或0x00194C。

时延，像带宽一样，也是一个静态特征的度量值，不需要动态地去量度。时延可以通过**show interface**命令显示的DLY参数来表示，单位是 μs （微秒）。一个接口的缺省时延可以通过命令**delay**进行更改，并以 $10\mu\text{s}$ 作为命令配置的最小计量单位。示例7-2显示了用**bandwidth**和**delay**命令更改示例7-1中的缺省带宽和时延的情况。

示例7-2 通过**bandwidth**和**delay**命令改变了接口e0的缺省度量值，使用**show interface**命令可以在输出端查看更改后的新度量值



在IGRP的更新消息中，时延也是用3个八位组字节来表示，并可以通过命令**delay**来改变，同样是以10 μ s作为最小计量单位。为了避免误解，这个数值用DLYIGRP来表示，以便区别于DLY，DLYIGRP可以通过命令**show interface**来观察，单位是 μ s（微秒）。例如，如果DLY的值是50，那么

$DLY_{IGRP} = DLY / 10 = 50 / 10 = 5$ ，或0x000005。

IGRP通过设定 $DLY_{IGRP} = 0xFFFFFFFF$ 来标识一条不可到达的路由，这个数值大约为167.8s，因此，一条IGRP路由端-端的最大时延是167s。

因为IGRP协议使用带宽和时延来作为它的缺省度量值，因此，这些特

性参数必须配置正确，并且要在所有的IGRP路由器的接口上统一规划配置。

除非有一个很好的理由，并且对更改这些参数的配置后所产生的结果有个清楚地理解，否则最好不要更改一个接口的带宽和时延参数。在大多数的情况下，最好保留使用这些参数的缺省值而不要加以改变。有一个值得注意的例外就是串行接口，正如本节前面所提及的，在Cisco路由器上，不管串行接口相连的是什么带宽的数据链路，它都会把串行接口的缺省带宽设置为1544。这时，应该使用命令**bandwidth**在接口上设置链路的实际带宽。

这里请注意，很重要的一点是在OSPF协议中也使用带宽来计算它的度量值。因此，在一个同时运行IGRP和OSPF协议的网络中，如果要改变IGRP的度量值，应该使用**delay** 来影响IGRP的度量值，如果改变带宽将会同时影响到IGRP和OSPF。

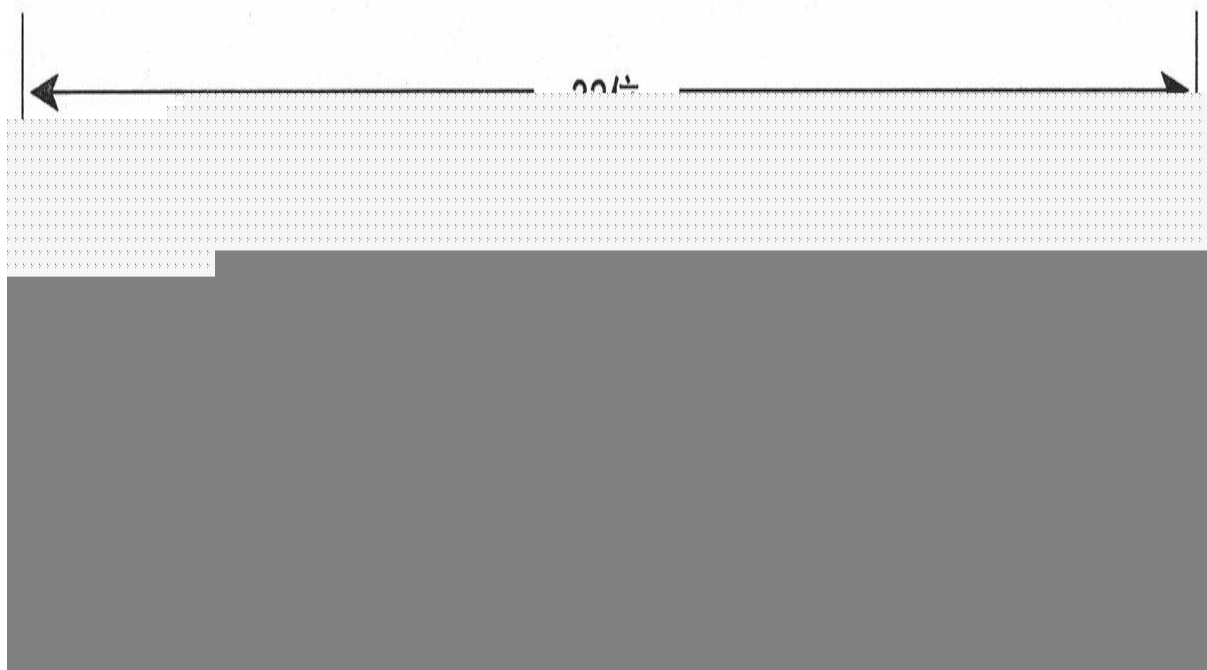
表7-1中列出了一些常用接口的带宽和时延（注意，串行接口的缺省带宽总是1544；表7-1显示了使用**bandwidth**命令来反映实际相连的链路带宽的效果）。

表7-1 常用BW_{IGRP}和DLY_{IGRP}的数值

2015

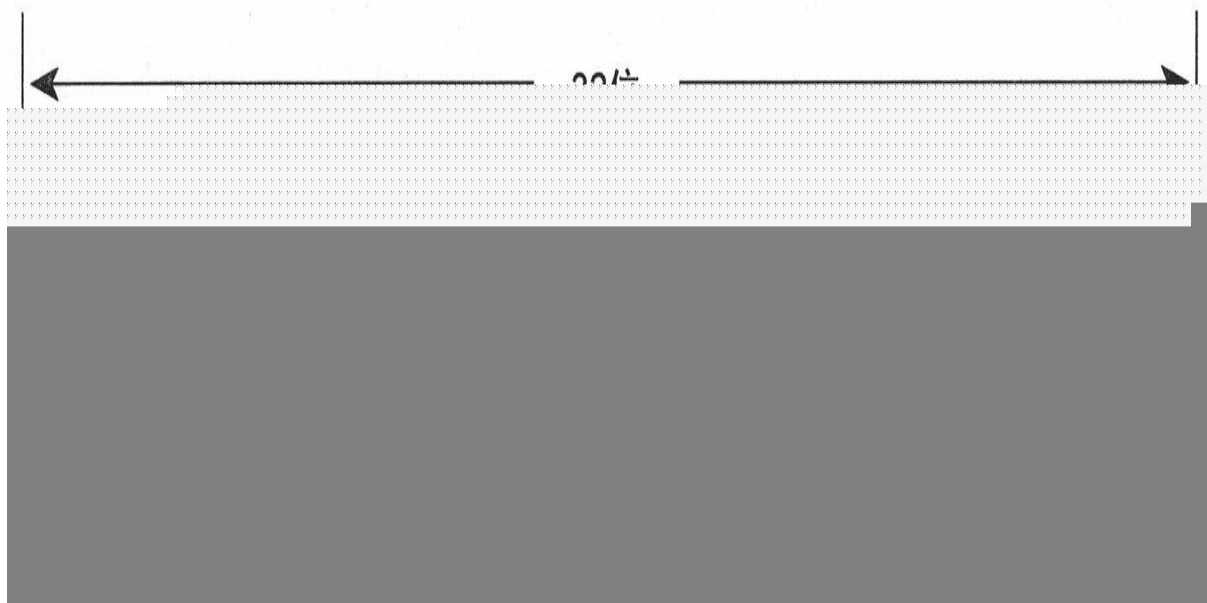
可靠性是一个动态的度量参数，它使用一个8位的数字来表示，255表示100%的可靠链路，而1表示最低可靠的链路。在命令**show interface** 的输出中，可靠性被表示成255的分数，例如，234/255（参见示例7-3）。

示例7-3 这个接口显示了该接口的可靠性是234/255，或91.8%



在IGRP的更新里，负载是一个8位的数字。在**show interface** 命令的输出中表示成一个255的分数，例如，40/255（参见示例7-4）；1表示最小的负载链路，255表示100%的负载链路。

示例7-4 这个接口显示了一个**40/255**的负载链路，或**15.7%**



如果可靠性或负载被用来作为一个度量或复合度量的一部分，计算度量

的算法不允许在出错率或信道占用率上突然发生变化而影响网络的不稳定。举一个例子来说明，如果在一个可能出现变化的网络上，突发的大流量可能会导致路由进入抑制（holddown）状态，而通信量的突然降低可能会触发一个更新。为了防止度量频繁的改变，可靠性和负载是基于5min时间常数的指数加权平均计算的，它们每5s被更新一次。

对于每个IGRP路由的复合度量，可以用下面的公式计算：

$$\text{metric} = [k1 \times BW_{\text{IGRP}(\min)} + (k2 \times BW_{\text{IGRP}(\min)}) / (256 - \text{LOAD}) + k3 \times DLY_{\text{IGRP}(\text{sum})}] \times [k5 / (\text{RELIABILITY} + k4)]$$

在这里， $BW_{\text{IGRP}(\min)}$ 是沿着路由路径到达目的网络的所有出站接口的 BW_{IGRP} 带宽中的最小值，而 $DLY_{\text{IGRP}(\text{sum})}$ 是这条路由路径 DLY_{IGRP} 时延的总和。

系数k1到k5是可配置的加权值，它们的缺省值是：
k1=k3=1, k2=k4=k5=0。这些缺省

值可以通过下面的命令来更改： [\[3\]](#)

metric weights tos k1 k2 k3 k4 k5 ³

如果k5被设置为0，则 $[k5 / (\text{RELIABILITY} + k4)]$ 这一项将不使用。 [\[4\]](#)

使用k1~k5给定的缺省值，IGRP的复合度量的计算公式将简化成缺省的度量：

$$\text{metric} = BW_{\text{IGRP}(\min)} + DLY_{\text{IGRP}(\text{sum})}。$$

在图7-3中的例子显示了网络的每个路由器接口的带宽和时延的配置，以及计算出IGRP路由度量的路由器之一的转发数据库。 [\[5\]](#)

路由选择表本身只显示了已经计算出来的度量，如果需要查看IGRP记录的每条路由实际的参数值，可以用**show ip route address** 命令，参见示例7-5。这里路由器Casablanca到子网172.20.40.0/24的路由路径上的最小

带宽是512kbit/s，最小带宽的链路接口位于路由器Quebec的出站接口。该路由时延的总和是（1000+20000+20000+5000）=46000μs。

$$BW_{IGRP (min)} = 10^7 / 512 = 19531$$

$$DLY_{IGRP (sum)} = 46000 / 10 = 4600$$

$$metric = BW_{IGRP (min)} + DLY_{IGRP (sum)} = 19531 + 4600 = 24131$$

示例7-5中显示了除了跳数之外IGRP还记录了沿着路由最小的MTU。MTU不用来作度量的计算。跳数是下一跳路由器报告的跳数，仅仅用来限制网络规模的大小。缺省条件下，最大的跳数是100，也可以通过命令**metric maximum-hops** 配置成1~255之间的数值。如果一条路由超过了设置的最大跳数，那么它的时延将被设置成0xFFFFF，而变成一条不可达的路由。

请注意，这里所有的度量都是基于沿着路由方向的路由器出站接口计算的。例如，路由器Yalta到达子网172.20.4.0/24的路由度量是不同于路由器Casablanca到达子网172.20.40.0/24的路由度量的。这是因为，路由器Yalta和路由器Quebec之间的链路带宽的配置不同，而且到达这两个目标子网的出站接口上时延也不同。

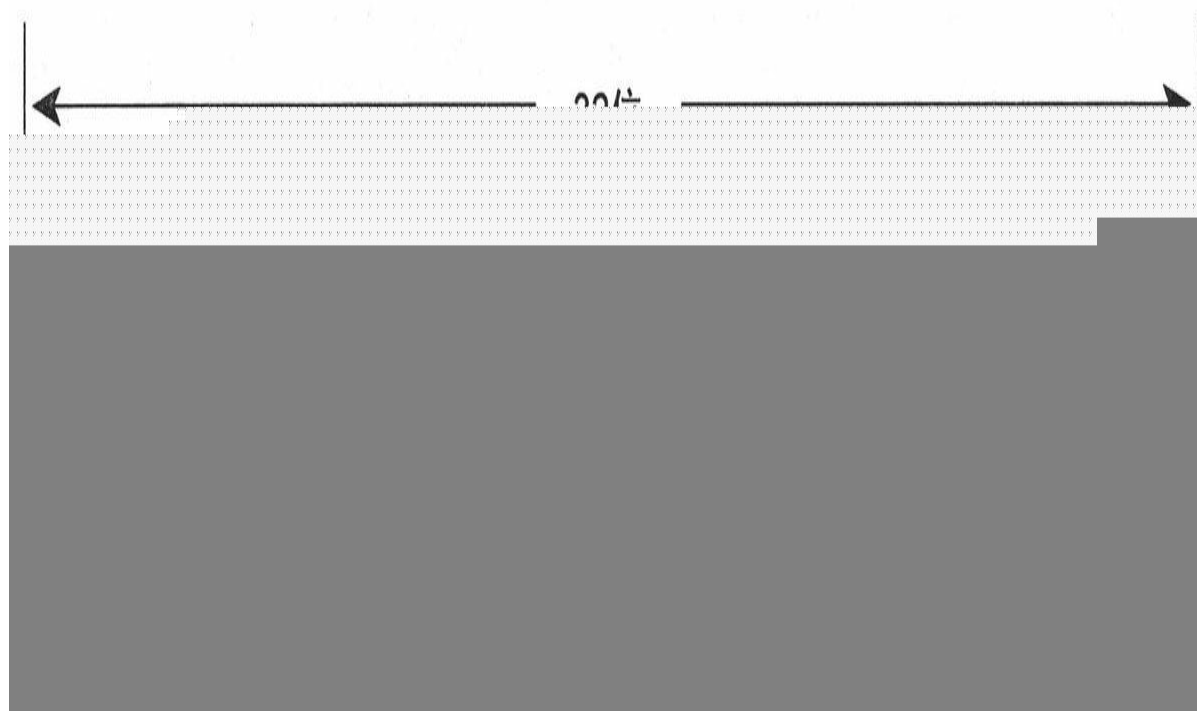


图7-3 缺省条件下，**IGRP**的度量是由时延的总和加上最小的带宽计算得来的

示例7-5 路由器**Casablanca**到子网**172.20.40.0/24**的路由的度量由**512kbit/s**的最小带宽和**46000 μ s**的路由时延总和计算得出



7.2 从IGRP到EIGRP

最初推动开发EIGRP的动机仅仅是简单的为了使IGRP支持无类别路由选择。但是，在协议开发的早期，工作于该开发项目的工程师们接受了一些有关一个新的收敛算法的学术建议，决定在IGRP协议的性能扩展项目中使用这个算法。结果，EIGRP协议除了保留IGRP协议引入的一些概念外，例如多种度量、协议域和非等价负载均衡，该协议和IGRP有明显的不同。

EIGRP协议有时也被描述成一个具有链路状态协议行为特性的距离矢量协议。在这里，对第4章中的广泛论述作一个扼要的重述：距离矢量协议是路由器之间共享路由器所知道的所有信息，但仅仅限于在与之直连的邻居之间共享；而链路状态协议虽然只通告它们直连链路的信息，但是链路状态协议可以在它们的路由选择域或区域内的所有路由器上共享这些信息。

到目前为止，所讨论的所有距离矢量协议的运行都是基于Bellman-Ford（或Ford-Fulkerson）算法或其一些派生算法的基础之上的。因此，这些协议易于产生路由选择环路和计数无穷大的问题。结果是，这些协议必须要采取一些避免路由选择环路的措施，像水平分隔、路由毒性逆转和抑制计时器等。由于每一台路由器在向它的邻居传送路由信息之前，都必须对收到的路由信息运行路由选择算法，因此大型网络的收敛可能会变得比较慢。更为重要的是，距离矢量协议在通告路由时，如果网络核心的关键链路发生了变化，就意味着有很多发生变化的路由要进行通告。

相对于距离矢量协议，链路状态协议受到路由选择环路和有害路由选择信息的影响就小得多了。首先，链路状态数据包的转发不依赖于路由计算的执行，因而大型网络就可以更快速地进行收敛。其次，它只通告链路和链路的状态，而不通告路由，这意味着链路的变化不会引起使用这条链路的所有路由都被通告。

其他的路由选择协议执行路由的计算——不论是在给它的邻居发送距离矢量更新之前，还是在生成网络拓扑数据库之后，它们的共同特性都是单独地进行路由的计算。相反，EIGRP协议使用了一个称为扩散计算（diffusing computations）的方法——在多台路由器之间通过一个并行

的方式执行路由的计算，从而在保持无环路的拓扑时可以随时获取较快的收敛。

虽然EIGRP更新仍然是将距离矢量传送给它直连的邻居，但是EIGRP更新是非周期的、部分的和有边界的。“非周期的”意思是指更新不是按照规是的时间间隔发送的，而是在度量或网络拓扑发生变化时才发送更新。“部分的”意思是指更新只包含发生变化的路由条目，而不是路由器的所有路由条目。“有边界的”意思是指更新仅仅发送给受到影响的路由器。这些特性意味着，EIGRP协议比典型的距离矢量协议所使用的带宽少得多，这个特点对路由选择协议在带宽较低而费用较高的广域网（WAN）链路上运行时显得特别重要。

另外一个需要关注的是，如果是在低带宽的广域网链路上进行路由选择，在路由收敛期间，路由选择信息的流量会显得比较大，这时要考虑可以使用的最大带宽。缺省情况下，EIGRP协议使用的带宽不超过链路总带宽的50%。后来IOS发布的版本允许使用命令**ipbandwidth-percent eigrp** 来改变这个缺省的百分比。

EIGRP协议是一个无类别的协议（也就是说，在其路由更新中的每一个路由条目都包含子网掩码）。EIGRP协议不仅可以利用可变长子网掩码进行子网的划分（如第6章中所讲述的），而且可以利用可变长子网掩码进行地址的聚合（address aggregation），即一组主网络地址的汇总。

EIGRP协议能够使用MD5加密校验和对EIGRP数据包进行认证。基本的认证和MD5认证已经在第6章中讲述过了；本章仅讲述一个配置EIGRP认证的实例。

最后，EIGRP协议的一个主要特性是它不仅可以进行IP协议的路由选择，而且可以进行IPX协议和AppleTalk协议的路由选择。

7.3 EIGRP的基本原理与实现

EIGRP协议使用和IGRP协议相同的公式来计算它的复合度量值。但是，EIGRP协议使用一个256的倍数因子扩展了度量参数，使它具有更好的度量粒度。因此，如果在一条到达目的地的路径上，配置的最小带宽是512kbit/s，并且配置的总延迟是46000μs，那么IGRP协议计算出的复合度量值应该是24131。而EIGRP协议将使用256倍数乘以带宽和延迟参数，从而计算出的度量值是 $256 \times 24131 = 6177536$ 。

如图7-4所示，EIGRP协议包含以下4个部件：

- 依赖于协议的模块；
- 可靠传输协议（RTP）；
- 邻居发现和恢复；
- 扩散更新算法（DUAL）。

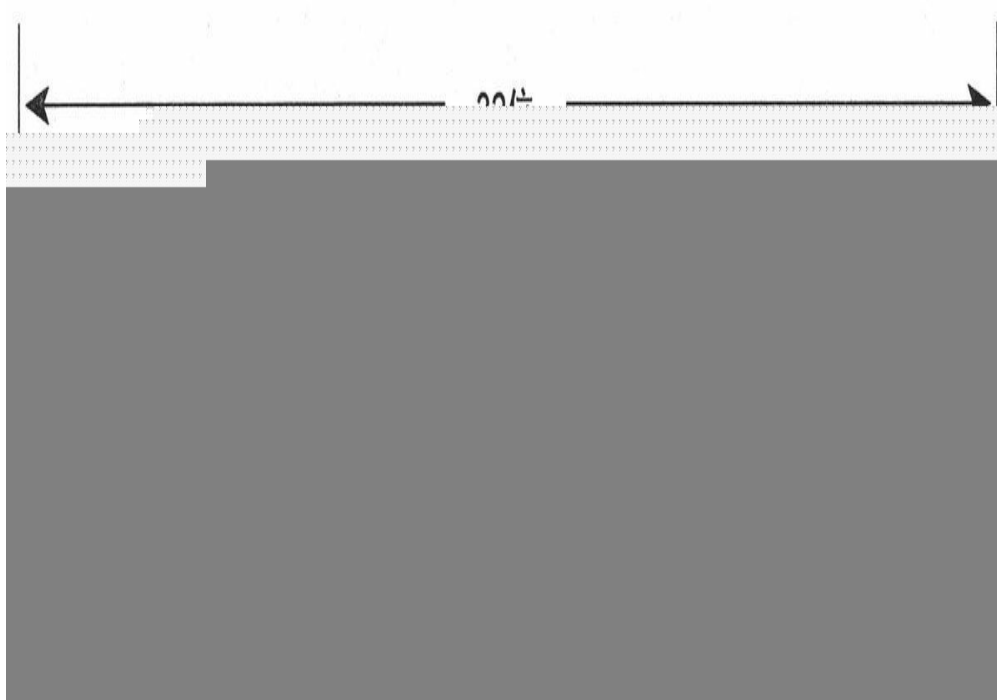


图7-4 EIGRP协议的4个主要部件。**RTP**和邻居发现是使**DUAL**正确

操作的更低层次上的协议。**DUAL**可以在多个可路由的协议上执行路由计算

本节将讲述EIGRP的每一个部件，并特别注重地讲解DUAL，最后再讨论地址聚合的内容。

7.3.1 依赖于协议的模块（Protocol-Dependent Modules）

EIGRP协议实现了IP协议、IPX协议和AppleTalk协议的模块，它可以担负起某一特定协议的路由选择任务。例如，IPX EIGRP模块可以负责在IPX网络上与其他IPX EIGRP进程进行路由信息交换，并且将这些信息传递给DUAL。另外，IPX模块也接收和发送SAP信息。

正如图7-4所示，每个单独模块的通信量被封装在它们各自的网络层协议中；例如，对于IPX协议的EIGRP通过IPX协议数据包传输。

EIGRP协议在很多情况下和其他路由选择协议自动进行路由重新分配：

- IPX EIGRP将自动地和IPX RIP协议、NLSP协议进行路由重新分配；
- AppleTalk EIGRP将自动地和AppleTalk RTMP协议进行路由重新分配；
- 如果IGRP进程和EIGRP进程在同一个自主系统内，那么IP EIGRP也将自动地和IGRP进行路由重新分配。

EIGRP和其他IP路由选择协议之间的路由重新分配将在第11章中讨论。

配置关于IPX和AppleTalk的EIGRP超出了本书的讲解范围，请参阅Cisco配置手册以获取更多的信息。

7.3.2 可靠传输协议

可靠传输协议（Reliable Transport Protocol, RTP）用来管理EIGRP数据包的发送和接收。可靠的发送是指发送是有保障的而且数据包是有序地发送的。有保障的发送是依赖Cisco公司私有的算法来实现的，这个私有的算法被称为“可靠组播（reliable multicast）”，它使用保留的D类地址

224.0.0.10。每一个接收可靠组播数据包的邻居都会发送一个单播的确认数据包。

有序的发送是通过在每个数据包中包含两个序列号来实现的。每一个数据包都包含一个由发送该数据包的路由器分配的序列号，这个序列号在每台路由器发送一个新的数据包时递增1。另外，发送路由器会把最近从目的路由器收到的数据包的序列号放在该数据包中。

在一些实例中，RTP也可以使用不可靠的发送，不需要确认，而且在使用不可靠发送的EIGRP数据包中不包含序列号。

EIGRP协议使用多种类型的数据包，所有这些数据包都通过IP头部的协议号88来标识。

- **Hello (Hello)** ——用于邻居发现和恢复进程。Hello数据包使用组播方式发送，而且使用不可靠的发送方式。
- **确认 (Acknowledgments, ACK)** ——是不包含数据的Hello数据包。ACK总是使用单播方式和不可靠的发送方式。
- **更新 (Update)** ——用于传递路由更新信息。不像RIP协议和IGRP协议的更新，EIGRP协议的这些更新数据包只在必要的时候传递必要的信息，而且仅仅传递给需要路由信息的路由器。当只有某一指定的路由器需要路由更新时，更新数据包就是单播发送的；当有多台路由器需要路由更新时，更新数据包就是组播发送的，例如，路由的度量和拓扑发生变化时。更新数据包总是使用可靠的发送方式。
- **查询 (Query) 和答复 (Reply)** ——是DUAL有限状态机 (DUAL finite state machine) 用来管理它的扩散计算的。查询消息可以使用组播方式或者单播方式发送，而回复消息总是单播方式发送的。查询和回复数据包都使用可靠的发送方式。
- **请求 (Request)** ——最初打算提供给路由服务器使用的数据包类型。但是这个应用从来没有实现过，在这里提到请求数据包主要是因为一些老的文档中可能会提及它们。

如果任何数据包通过可靠的方式组播出去，而没有从邻居那里收到一个

ACK数据包，那么这个数据包就会以单播方式被重新发送给那个没有响应的邻居。如果经过16次这样的单播重传还没有收到一个ACK数据包的话，那么这个邻居就会被宣告为无效。

在从组播方式切换到单播方式之前等待一个ACK时间可以由组播流计时器（multicast flow timer）指定。后续的单播之间的时间可以由重传超时（Retransmission Timeout, RTO）指定。对于每一个邻居，组播流计时器和重传超时都可以通过平均回程时间（Smooth Round-Trip Time, SRTT）来计算。SRTT是一个用来衡量路由器发送EIGRP数据包到邻居和从邻居那里接收到该数据包的确认所花费的平均时间，以毫秒（ms）为单位。关于SRTT、RTO和组播流计时器的精确值的计算公式是私有版权的。

下面两小节将讲述使用不同数据包类型的EIGRP的部件。

7.3.3 邻居发现和恢复

因为EIGRP协议的更新消息是非周期的，因此有一个发现和跟踪邻居的方法是非常重要的；在这里，邻居是指网络上直连的通告EIGRP的路由器。在大多数网络中，Hello数据包是以组播方式每5s发送一次的，其中减掉一个很小的随机时间差用来防止更新的同步。在多点的X.25、帧中继和ATM接口上，由于它们的接入链路速率通常是T1或更低的速率，因此它们的Hello数据包是以单播方式每60s发送一次的。[\[6\]](#)这个比较长的Hello数据包时间间隔也缺省地使用在ATM SVC和ISDN PRI的接口上。在所有的实例中，Hello数据包都是不进行确认的。缺省的Hello数据包的时间间隔可以在每个接口上使用命令`ip hello-interval eigrp`进行更改。

当一台路由器从它的邻居路由器收到一个Hello数据包时，这个数据包将包含一个抑制时间（holdtime）。这个抑制时间会告诉本路由器，在它收到后续的Hello数据包之前等待的最长时间。如果抑制计时器超时了，路由器还没有收到Hello数据包，那么将宣告这个邻居不可到达，并且通知DUAL这个邻居丢失了。在缺省的情况下，抑制时间是Hello时间间隔的3倍，也就是说，对于低速的非广播多路访问（NBMA）网络来说是180s，对于其他所有的网络来说是15s。这个缺省值可以通告在每个接口上配置命令`ip hold-time eigrp`来更改。EIGRP协议具有在15s以内检测邻居丢失的能力，对照RIP协议的180s和IGRP协议的270s所花费

的时间，显然这是一个对EIGRP的快速收敛起很大作用的因素。

每一个邻居的相关信息都记录在一个邻居表中。参见示例7-6所示，邻居表（neighbor table）记录了邻居路由器的IP地址和收到邻居Hello数据包的接口。邻居通告的抑制时间作为SRTT和邻居关系建立时间

（uptime）也记录在邻居表中，这里的邻居关系建立时间是指从邻居第一次被添加到邻居表后到现在所经过的时间。重传超时（RTO）是指在一个组播方式的数据包发送失败后，路由器等待一个单播方式发送的数据包的确认时间，单位是毫秒（ms）。如果一个EIGRP的更新、查询或答复数据包被发送出去，那么这个数据包的一个拷贝就会放在一个重传队列里排队。如果重传超时了还没有收到确认数据包，那么重传队列中数据包的另一个拷贝将被再次发送出去。队列计数（Q Count）就是标识在这个重传队列中等待发送的数据包数量的。从邻居收到的最新的更新、查询或答复数据包的序列号也记录在了邻居表中。可靠传输协议RTP跟踪这些序列号，以确保来自邻居的数据包不是无序的。最后，H列记录了这台路由器所学到的邻居的顺序号。

示例7-6 show ip eigrp neighbors命令用来观察IP EIGRP的邻居表



7.3.4 扩散更新算法

扩散更新算法（Diffusing Update Algorithm, DUAL）是一个收敛算法，它代替了用于其他距离向量协议使用的Bellman-Ford或Ford-Fulkerson算法。DUAL算法背后的设计思想是，即使暂时的路由选择环路也会对一个网络的性能造成损害。DUAL最初是由E. W. Dijkstra和C.S.Scholten提议的[7]，指的是为了随时能够打破路由环路，而使用扩散计算去执行

一个分布式最短路径路由选择。虽然很多研究人员对DUAL算法的发展作出了贡献，但是最显著的贡献来自于J. J. Garcia-Luna-Aceves的工作。[\[8\]](#)

1. DUAL: 预备概念

为了能够正确地操作DUAL，较低层的协议必须确保满足下面的几个条件：[\[9\]](#)

- 一个节点需要在有限的时间内检测到一个新邻居的存在或一个相连邻居的丢失。
- 在一个正在运行的链路上传送的所有消息应该在一个有限的时间内正确地收到，并且包含正确的序列号。
- 所有的消息，包括改变链路的代价、链路失败和发现新邻居的通告，都应该在一个有限的时间内一次一个地处理，并且应该被有顺序地检测到。

Cisco的EIGRP协议使用邻居发现/恢复和可靠传输协议（RTP）来确定这些前提条件。在介绍DUAL之前，先来介绍几个术语和概念。

（1）邻接（adjacency）

刚启动时，路由器使用Hello数据包发现它的邻居并标识自己给邻居识别。当邻居被发现时，EIGRP协议将试图和它的邻居形成一个邻接关系。邻接是指两个互相交换路由信息的邻居之间形成的一条逻辑的关联关系。一旦邻接成功地建立，路由器就可以从它们的邻居那里接收路由更新消息了。这里的路由更新消息包括发送路由器所知道的所有路由和这些路由的度量值。对于每一条路由，路由器都将会基于它邻居通告的距离（distance）和到它的邻居的链路代价计算出一个距离。

（2）可行距离（Feasible Distance, FD）

到达每一个目的地的最小度量将作为该目的网络的可行距离。例如，路由器可能得到3条到达子网172.16.5.0不同的路由，这3条路由计算所得的度量分别是380672、12381440和660868; 380672为可行距离（FD），因为它是经计算到达子网172.16.5.0的最小度量。

（3）可行性条件（Feasibility Condition, FC）

可行性条件就是需要满足下面这样的条件——本地路由器的一个邻居路由器所通告的到达一个目的网络的距离是否小于本地路由器到达相同目的网络的可行距离（FD）。

（4）可行后继路由器（Feasible Successor, FS）

如果本地路由器的邻居路由器所通告的到达目的网络的距离满足了FC，那么这个邻居就会成为该目的网络的一个可行后继路由器。[\[10\]](#)例如，假定一台路由器到达子网172.16.5.0的可行距离是380672，而它的邻居路由器所通告的到达该目的子网的路由路径的距离是355072，那么这台邻居路由器就满足FC，因而成为一台可行后继路由器；如果邻居路由器所通告的距离是380928，那么这台邻居路由器就不满足FC，因而也就不能成为一台可行后继路由器。

可行后继路由器和可行性条件的概念是避免环路的一项核心技术，因为可行后继路由器总是“下游路由器（downstream）”（也就是说，可行后继路由器到达目的地的度量距离比本地路由器的可行距离（FD）更短），所以路由器从来不会选择一条导致反过来还要经过它本身的路径——像这样的路径一般有一个大于本地路由器FD的距离。

存在一个或多个可行后继路由器的每一个目的网络，将与下面的每一项一起被记录在一个称为“拓扑结构表（topological table）”的表中：

- 目的网络的可行距离；
- 所有的可行后继路由器；
- 每一个可行后继路由器所通告的到达目的网络的通告距离；
- 本地路由器所计算的经过每一个可行后继路由器到达目的网络的距离，也就是基于可行后继路由器所通告的到达目的子网的距离和本地路由器与该可行后继路由器之间相连链路的成本计算所得的距离；
- 与发现每一个可行后继路由器的网络相连的接口。[\[11\]](#)

（5）后继路由器（successor）

对于在拓扑结构表中列出的每一个目的网络，将选用拥有最小度量值的路由并放置到路由表中。通告这条路由的邻居就成为一个后继路由器，或者是到达目的网络的数据包的下一跳路由器。

举一个例子可以帮助我们澄清这些术语，但首先简要地描述一下本小节这个例子中用到的网络还是必要的。图7-5显示了一个基于EIGRP的网络，这个网络将在本小节和后续的3个小节中使用。[\[12\]](#)把命令**metric weights 0 0 0 1 0 0**添加到EIGRP的进程中，因此只使用延迟来进行路由的度量计算。命令**delay** 使用图中每一个链路上标注的数值。例如，路由器Wright和路由器Langley的接口和子网10.1.3.0相连，那么接口配置的延迟就是2。这些做法将给下面的例子带来方便和简化。

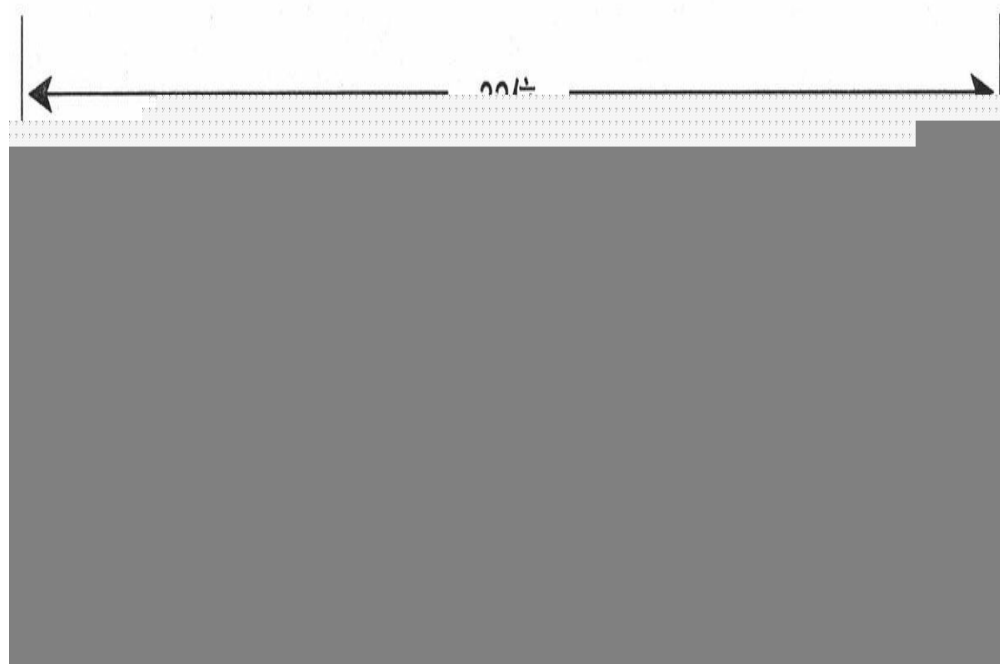


图7-5 本小节和后续的两个小节的图例和例子都是基于**EIGRP**协议的网络

需要指出的是，虽然这里使用的延迟参数为了简化而不太实用，但是进行度量的方法还是很实用的。很多参数是通过接口命令**bandwidth** 所指定的带宽进行计算的。如命令**ip bandwidth-percent eigrp** 就是直接应用于EIGRP的；但OSPF的代价并不是直接地应用于OSPF的。因此，除了需要设置串行链路的带宽与其实际带宽相符，还应该避免改变带宽的配

置。如果需要改变一个接口的度量来影响EIGRP（或IGRP协议）的路由选择，还应该使用命令**delay**。这样可以避免很多令人头痛的问题发生。

在示例7-7中，可以使用命令**show ip eigrp topology** 来查看路由器Langley的拓扑结构表。图7-5中显示的7个子网的每个子网和这些子网的可行后继路由器，都列在拓扑结构表中。例如，子网10.1.6.0的可行后继路由器10.1.3.1（路由器Wright）和10.1.5.2（路由器Chanute），分别通过接口S0和S1到达。

示例7-7 路由器Langley的拓扑结构表

Destination	Feasible Successor	Metric	Next Hop
10.1.6.0	10.1.3.1 (Wright)	1024	S0
10.1.6.0	10.1.5.2 (Chanute)	512	S1
10.1.5.0	10.1.3.1 (Wright)	1024	S0
10.1.5.0	10.1.5.2 (Chanute)	512	S1
10.1.4.0	10.1.3.1 (Wright)	1024	S0
10.1.4.0	10.1.5.2 (Chanute)	512	S1
10.1.3.0	10.1.3.1 (Wright)	1024	S0
10.1.3.0	10.1.5.2 (Chanute)	512	S1
10.1.2.0	10.1.3.1 (Wright)	1024	S0
10.1.2.0	10.1.5.2 (Chanute)	512	S1
10.1.1.0	10.1.3.1 (Wright)	1024	S0
10.1.1.0	10.1.5.2 (Chanute)	512	S1
10.1.0.0	10.1.3.1 (Wright)	1024	S0
10.1.0.0	10.1.5.2 (Chanute)	512	S1

圆括号中的两个度量也都是和每个可行后继路由器相关联的。第一个数字是本地路由器计算得出的路由器Langley到达目的网络的度量值。第二个数字是邻居通告的度量值。例如，图7-5中路由器Langley经过路由器Wright到达子网10.1.6.0的度量值是 $256 \times (2+1+1) = 1024$ ，而邻居路由器Wright通告的到达这个目的子网的度量值是 $256 \times (1+1) = 512$ 。通

过路由器Chanute到达上面相同目的子网的这两个度量值分别是
 $256 \times (4+1+1+1) = 1792$ 和 $256 \times (1+1+1) = 768$ 。

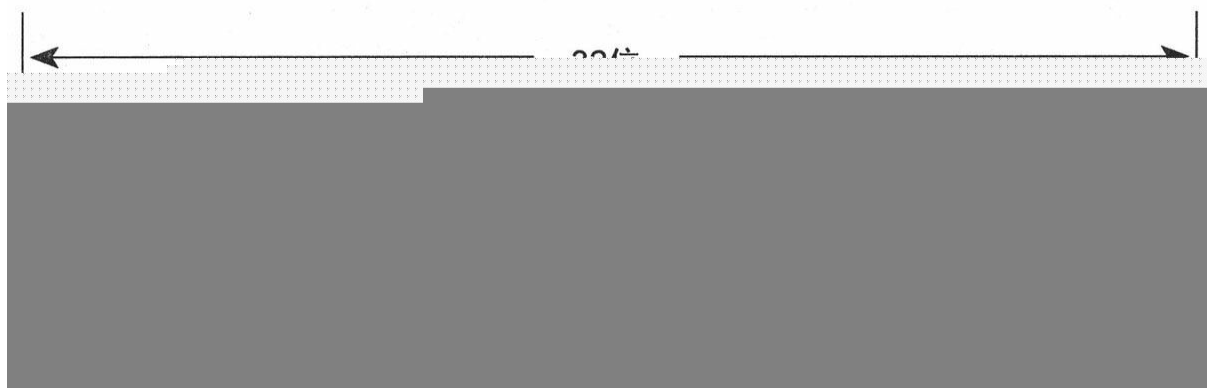
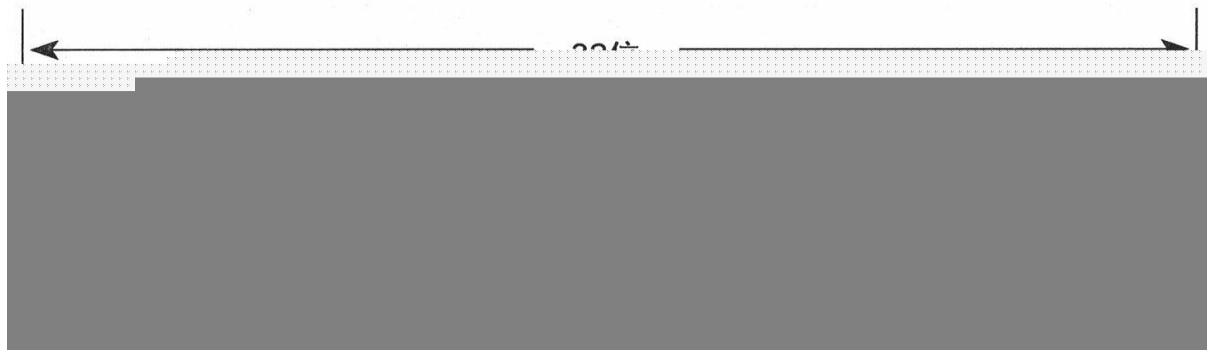
可见，从路由器Langley到达子网10.1.6.0的最小度量值是1024，因此，这个度量值就是可行距离（FD）。示例7-8中显示了路由器Langley的路由表，可以从中看出所选用的后继路由器。

示例7-8 基于最小的度量距离计算，路由器Langley的路由表显示了每个可行的目的网络选用的单台后继路由器

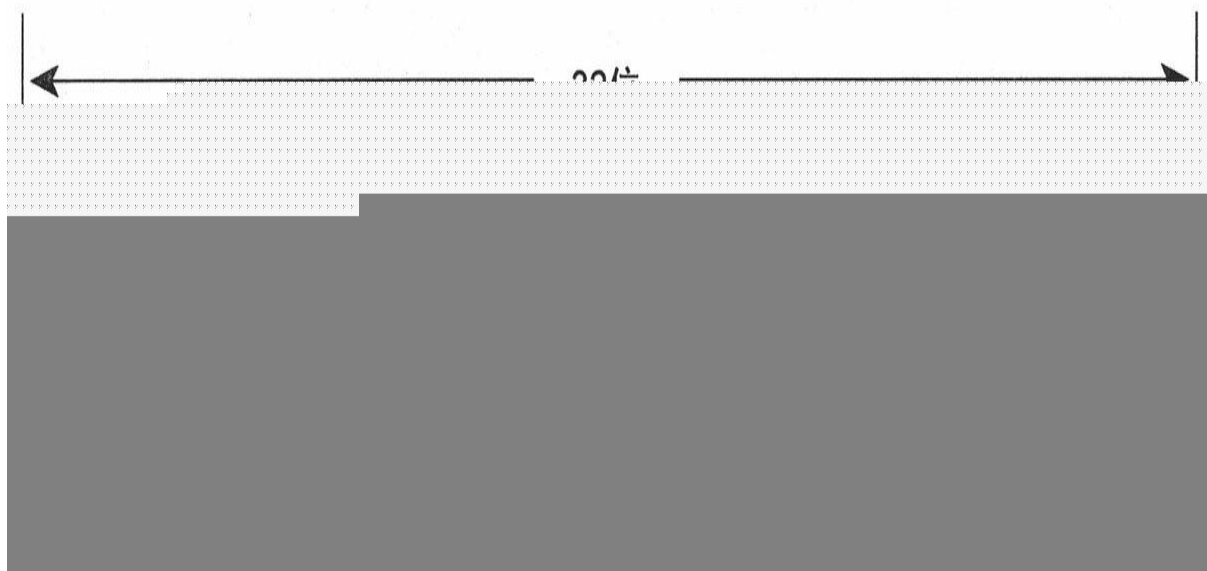


对于路由器Langley的每一条路由，只有一台后继路由器，而示例7-9中路由器Cayley的拓扑结构表却显示了到达目的子网10.1.4.0的两台后继路由器。这是因为在路由器Cayley所计算的度量值中，有两条路由的度量值都匹配它的可行距离。因此这两条路由都存在于示例7-10中的路由表中，并且路由器Cayley执行等价负载均衡。

示例7-9 路由器Cayley的路由拓扑结构表显示了到达目的子网10.1.4.0的两台后继路由器



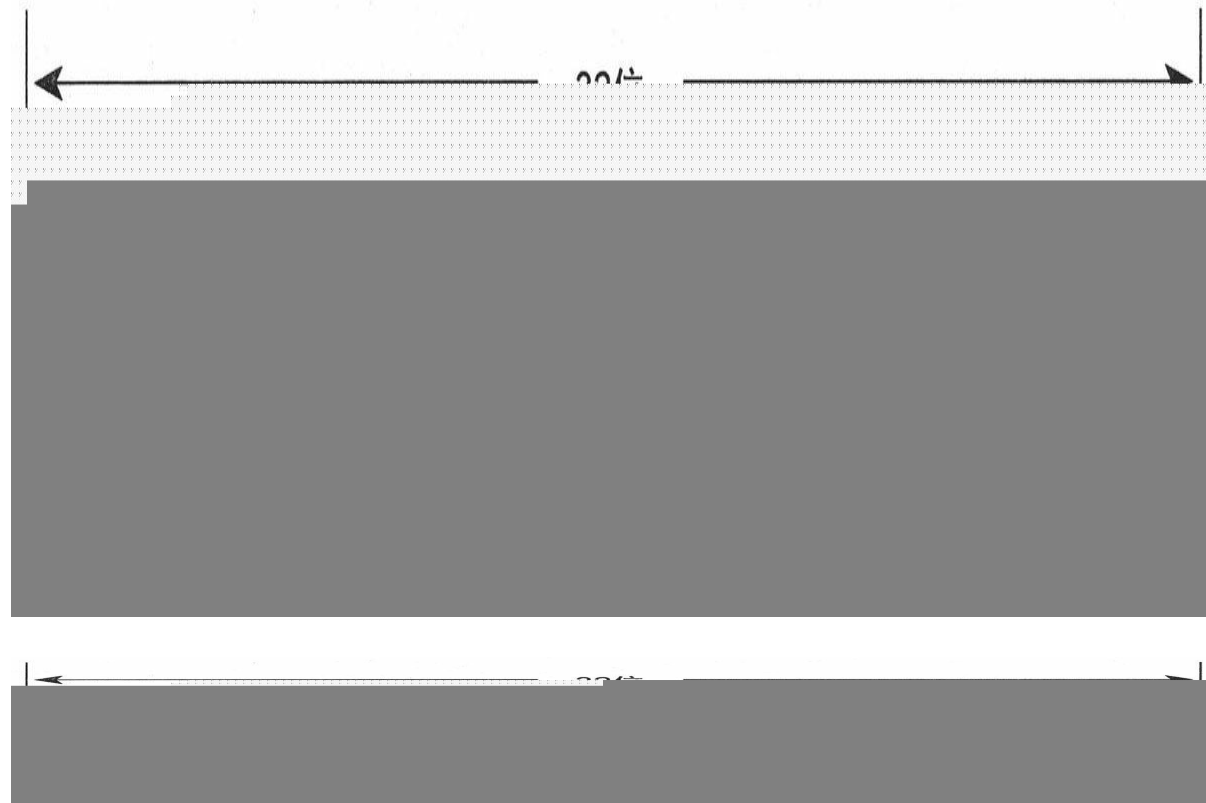
示例7-10 在到达目的子网**10.1.4.0**的两台后继路由器之间将执行等价负载均衡



示例7-11中的路由器Chanute的拓扑结构表显示了几条仅仅拥有一台可行后继路由器的路由。例如，到达子网10.1.6.0的路由拥有一个距离为768的可行距离（FD），因而路由器Wright（10.1.2.1）是惟一的可行后继

路由器。路由器Langley有一条到达子网10.1.6.0的路由，但是它的度量值是 $256 \times (2+1+1) = 1024$ ，大于FD，因此，路由器Langley到达子网10.1.6.0的路由不满足可行性条件（FC），因而路由器Langley也就没有资格成为一台可行后继路由器。

示例7-11 从路由器Chanute可达的几个子网只有惟一的一台可行后继路由器

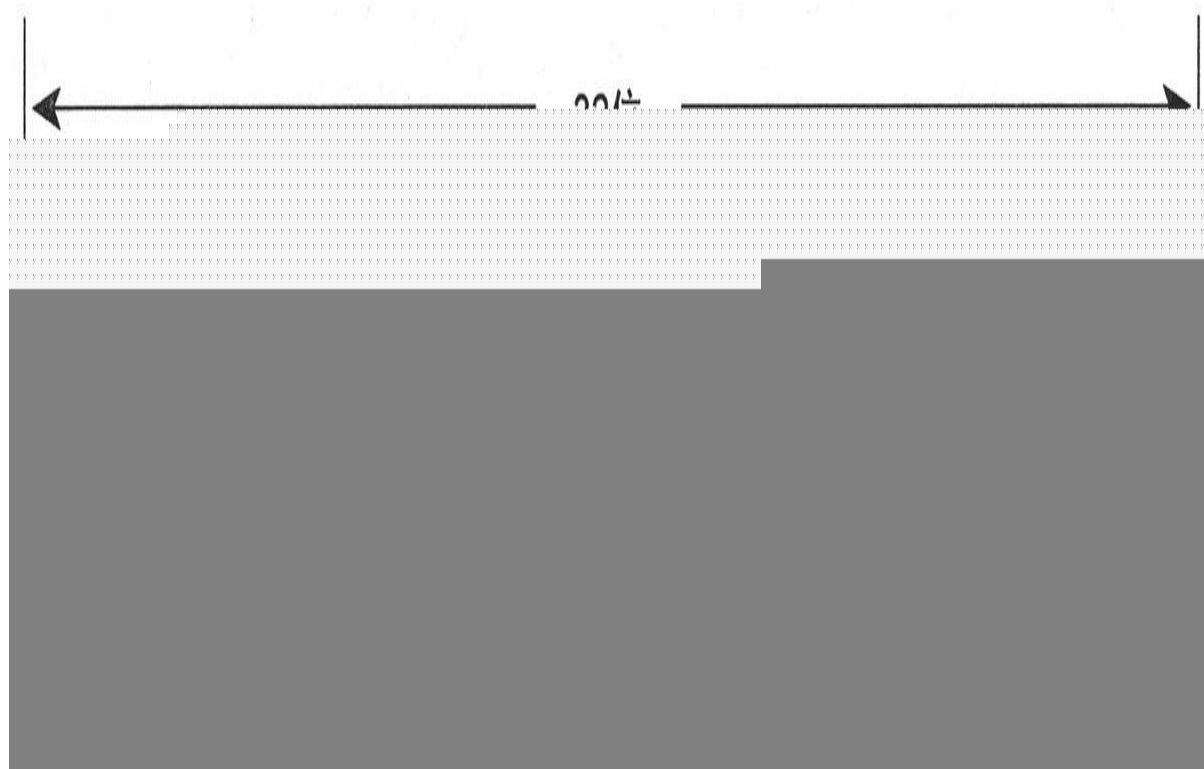


如果一台可行后继路由器通告的一条路由在本地路由器上所计算的度量比当前后继路由器的度量小，那么这台可行后继路由器就成为后继路由器。下面的情况可能会引起这种情形的发生：

- 发现一条新的路由；
- 一条后继路由器路由的度量值增加后超过了可行后继路由器的度量值；
- 一条可行后继路由器路由的度量值减小后小于后继路由器的度量值。

例如，示例7-12中显示了路由器Lilienthal到达子网10.1.3.0的后继路由器是路由器Cayley（10.1.6.2）。假设路由器Lilienthal和路由器Wright之间的链路代价减小到1，那么由于路由器Wright（10.1.4.1）正在通告一条到达子网10.1.3.0的距离是512的路由，同时根据路由器Lilienthal和路由器Wright之间新的链路代价值，路由器Lilienthal计算出经过路由器Wright到达目的子网的度量现在变成了768。因此，路由器Wright将替代路由器Cayley成为到达子网10.1.3.0的后继路由器。

示例7-12 路由器Lilienthal的拓扑结构表



其次，我们假定路由器Lilienthal发现了一个新的邻居，并且这个邻居正在通告一条到达子网10.1.3.0距离为256的路由。由于这个距离小于当前的可行距离（FD），因而这个新的邻居就成为一台可行后继路由器。进一步假定路由器Lilienthal到达这个新邻居的链路代价是256，那么路由器Lilienthal将计算得出经过这个新邻居到达子网10.1.3.0的度量值为512。这个度量值小于经过路由器Wright的度量值，因此这台新的邻居路由器将成为到达子网10.1.3.0的后继路由器。

由于可行后继路由器减少了扩散计算的数量，并提高了网络的性能，因此可行后继路由器十分重要。可行后继路由器也对降低重新收敛的次数

有一定的贡献。如果到达后继路由器的一条链路失效了，或者链路的代价增加并超过了可行距离（FD），那么这台路由器将首先在它的拓扑结构表中查找可行后继路由器，如果发现存在一台可行后继路由器，那么它就成为后继路由器；这种选择通常发生在次秒级范围内。路由器只有在找不到任何一台可行后继路由器的情况下，才开始进行扩散计算。成功进行EIGRP设计的关键在于，确保对所有的目的地来说总是存在一台可行后继路由器。

下面小节将给出一个更正式的规则集合，用来说明路由器是何时以及如何查找可行后继路由器的。这个规则集合称为DUAL有限状态机。

2. DUAL有限状态机

当一个EIGRP的路由器不执行扩散计算时，每一条路由都处于被动状态（passive state）。参见前面部分出现的所有的拓扑结构表，每一条路由左边的关键字就是用来指出路由的被动状态的。

在产生输入事件（input event）的任何时候，路由器都会重新评估一条路由的可行后继路由器的列表，这将在最后小节中讲述。一个输入事件可以是：

- 直连链路的代价发生变化；
- 直连链路的状态（up或down）发生变化；
- 收到一个更新数据包；
- 收到一个查询数据包；
- 收到一个答复数据包。

路由器重新评估的第一步是，在本地路由器上执行一个本地计算（local computation），也就是对于所有的可行后继路由器，重新计算到达目的地的距离。可能的结果有下面几种：

- 如果拥有最低的度量距离的可行后继路由器和已经存在的后继路由器不同，那么可行后继路由器将成为后继路由器；

- 如果新的度量距离小于FD，那么就更新FD；
- 如果新的度量距离和已经存在的度量距离不同，那么将向所有的邻居发送更新。

当路由器执行一个本地计算时，路由依然保持被动状态。如果本地路由器发现了一台可行后继路由器，那么将发送一个更新消息给它所有的邻居，但不改变路由的状态。

如果在拓扑结构表中没有发现任何一台可行后继路由器的话，那么路由器将开始进行扩散计算，而且路由器的路由状态改变成活动状态（**active state**）。在扩散计算完成和路由的状态返回到被动状态之前，路由器不能：

- 改变路由的后继路由器；
- 改变正在通告的路由的距离；
- 改变路由的FD；
- 开始进行路由的另一个扩散计算。

如图7-6所示，路由器是通过向它所有的邻居发送查询来开始一个扩散计算的，查询中包含一个到达目的地的新的本地路由器计算的距离。收到查询后，每一台邻居路由器将执行它自己的本地计算：

- 如果该邻居拥有到达目的地的一台或多台可行后继路由器，它将发送一个答复给原来发送查询的路由器。答复中将包含这台邻居路由器所计算的它到达目网络的最小距离。
- 如果一个邻居没有可行后继路由器，它将把路由的状态改变为活动状态，并且开始进行扩散计算。

对于每一台接收查询的邻居路由器，本地路由器将设置一个答复状态标记（**reply status flag (r)**）来不断地跟踪所有未处理的查询。当本地路由器收到所有发送到邻居路由器的查询的答复时，扩散计算就完成了。

在一些实例中，路由器没有收到发出的每一个查询的答复。例如，这有可能发生在一个拥有很多低速带宽或质量较差的链路的大型网络当中。

在扩散计算的开始，一个活动计时器（active timer）被设置为3min。

[13] 如果在活动计时器计时超时后还没有收到希望收到的所有答复，那么这条路由就被宣告“卡”在活动状态（Stuck-In-Active, SIA）。这些没有答复的邻居将从邻居表中删除，并且扩散计算认为这个邻居回应了一个无穷大的度量。

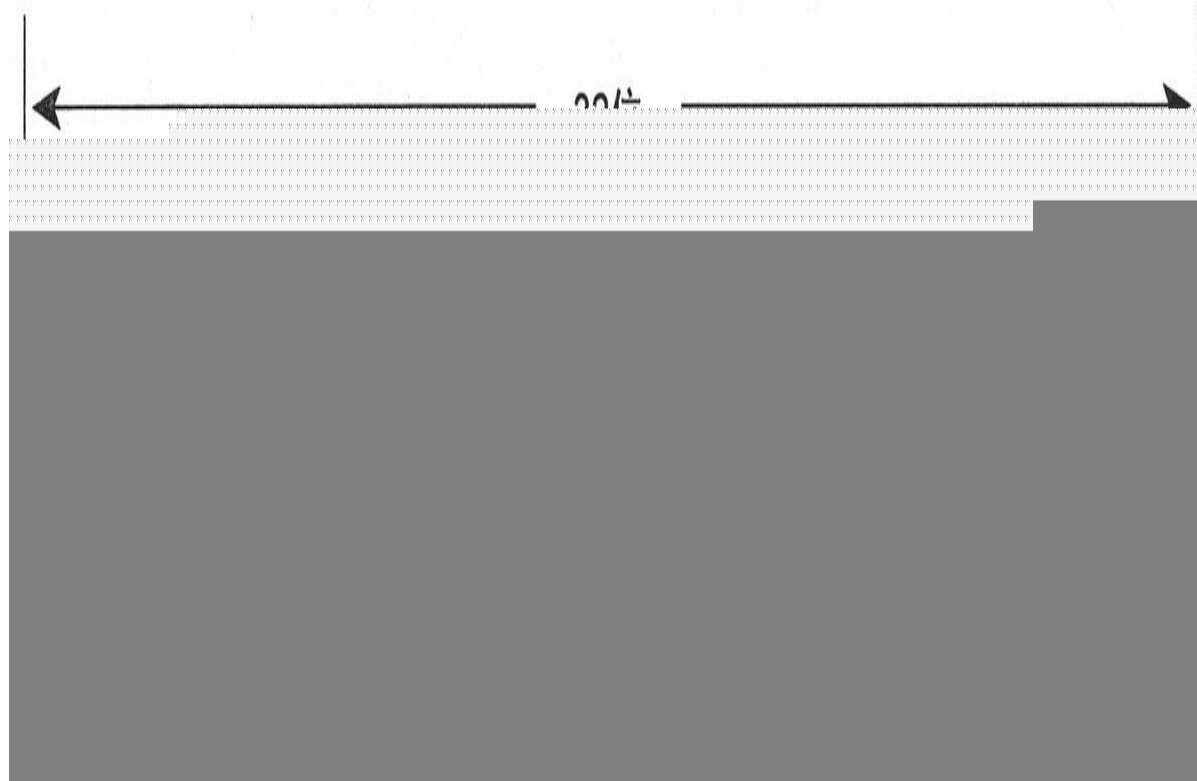


图7-6 扩散计算在查询被发送时扩大，而在答复被收到时收缩

活动计时器的缺省配置是3min，这个时长可以通过命令timers active-time来改变或使其无效。由于查询的丢失而造成邻居的删除显然会带来负面的影响，但是，在一个稳定的、设计良好的网络中，SIA的情形是从来不会发生的。本章的故障诊断一节中将更详细地讨论SIA及SIA过程最近的一个增强特性，即使用两个新的EIGRP消息：SIA Query和SIA Reply消息。这两个消息可以减少SIA故障的机会，也可以在发生故障时，使它们的故障诊断变得容易。

在扩散计算完成时，始发路由器会将FD设置成无穷大，这样可以确保任何答复到达目的地时有限距离的邻居路由器都满足FC，并成为一台可行后继路由器。对于这些答复消息，度量都是由答复消息中所通告的

距离加上与发送答复的邻居路由器相连的链路代价计算得出的。选择一台后继路由器是基于最低的度量值的，而且该度量值被设置为FD。任何一台可行后继路由器如果不满足该新的可行距离（FD）的可行性条件（FC）的话，就会从拓扑结构表中被删除。注意，在收到所有的答复之前不会选择后继路由器。

因为有多种类型的输入事件（input events）能够引起一条路由改变它的状态，所以当一条路由处于活动状态时就说明可能发生了一些类型的输入事件。DUAL定义了多种活动状态。查询始发标记（query origin flag（0））用来指出当前的状态。图7-7和表7-2中显示了完整的DUAL有限状态机。

有两个例子可以帮助我们阐明DUAL的处理过程。图7-8中显示了例子的网络拓扑，这里只需要关注到达子网10.1.7.0的每一台路由器的路径；参考图7-5中指定的地址。在数据链路上，箭头指出了每一台路由器用来到达子网10.1.7.0的后继路由器。在圆括号中显示的分别是每一台路由器到达子网的本地计算距离、路由器的FD、答复状态标记（reply status flag（r））和查询始发标记（query origin flag（O））。在后面使用这个网络的图示和例子中，活动路由器（active router）都用一个圆圈指出。

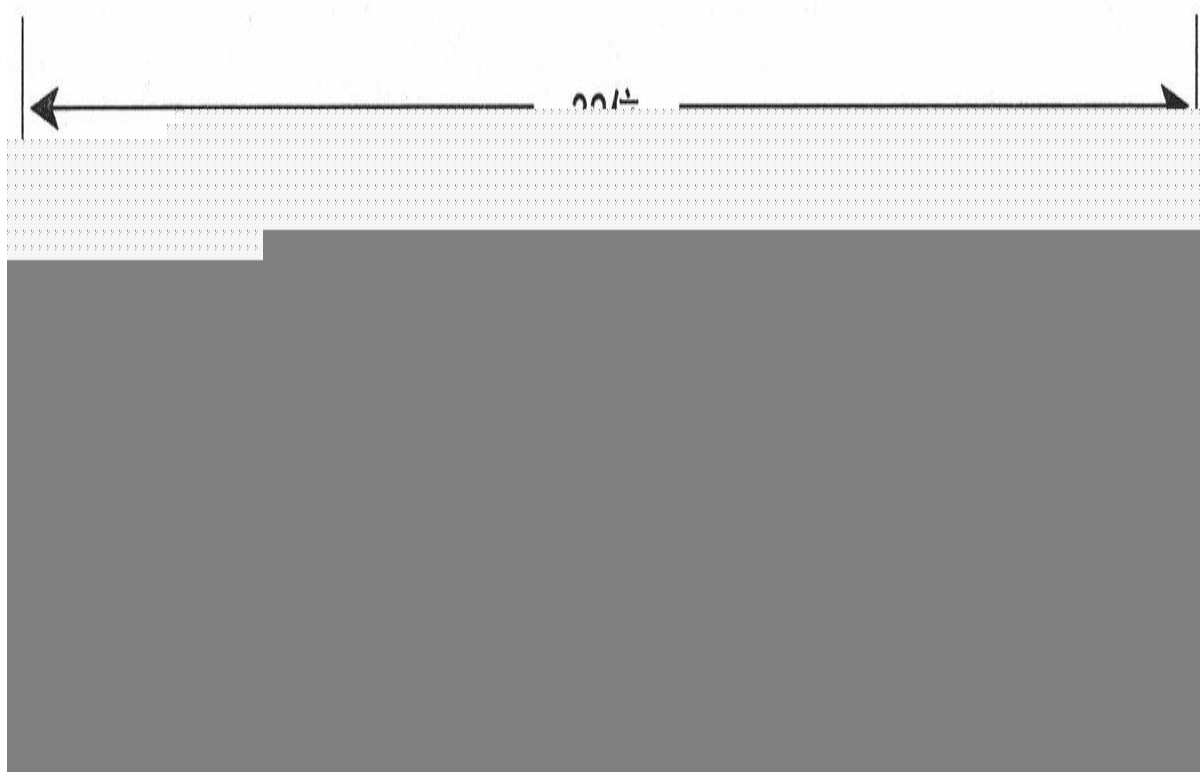


图7-7 DUAL有限状态机。查询始发标记（**query origin flag (0)**）标识出扩散计算当前的状态。参看表7-2关于每一个输入事件（**IE**）的解释

表7-2 DUAL有限状态机的输入事件

输入事件	描述
IE1	满足FC或者目的地不可到达的任何输入事件
IE2	从后继路由器收到了查询消息；不满足FC
IE3	除了来自后继路由器查询的其他输入事件；不满足FC
IE4	除了最新的答复或来自后继路由器的查询的输入事件
IE5	除了最新的答复、来自后继路由器的查询或到达目的地距离的增加的输入事件
IE6	除了最新答复的输入事件；
IE7	除了最新答复或到达目的地距离的增加的输入事件
IE8	到达目的地距离的增加
IE9	收到了最新的答复；当前FD不满足FC

- IE10 从后继路由器收到了查询
- IE11 收到了最新的答复；FC和当前FD匹配
- IE12 收到了最新的答复；设置FD为无限大

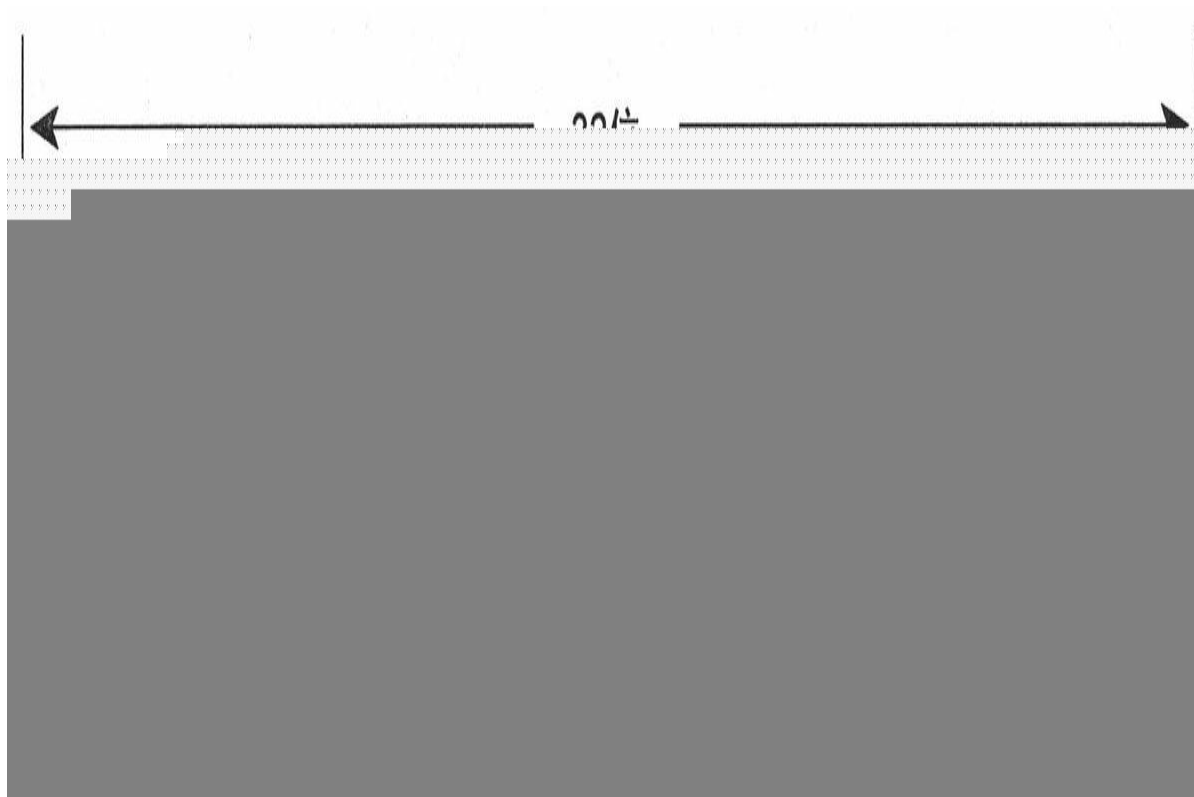


图7-8 到达子网10.1.7.0的所有路由都是被动状态的，通过r=0和0=1指示

3. 扩散计算：范例1

这个例子关注的是路由器Cayley和它到达子网10.1.7.0的路由。在图7-9中，路由器Cayley和Wright（10.1.1.1）之间的链路是失效的。EIGRP把失效的链路当作一条拥有无穷大距离的链路。[\[14\]](#)路由器Cayley检查它的拓扑结构表，来查找到达子网10.1.7.0的可行后继路由器，但是没有找到，参见示例7-9所示。

如图7-10所示，路由器Cayley的路由变成了活动状态。这条路由的距离和FD也变为不可到达的了，并且路由器Cayley把一个包含新距离的查询发送给它的邻居路由器Lilienthal。路由器Cayley关于路由器Lilienthal的

答复状态标记被设置为1，用来指出期待一个答复，因此输入事件是一个查询的未接受状态（IE3）， $0=1$ 。

一旦收到了查询消息，路由器Lilienthal将执行一个本地计算，如图7-11所示。参见示例7-12，由于路由器Lilienthal拥有到达子网10.1.7.0的可行后继路由器，因而它的路由将不会变为活动状态。路由器Wright成为新的后继路由器，而且路由器Lilienthal将发送一个含有它经过路由器Wright到达子网10.1.7.0的距离的答复消息。因为到达子网10.1.7.0的距离已经增加了，而且路由不处在活动状态，因此路由器Lilienthal的FD没有变化。



图7-9 路由器Cayley和Wright间的链路是失效的，因而路由器Cayley没有一台到达子网10.1.7.0的可行后继路由器

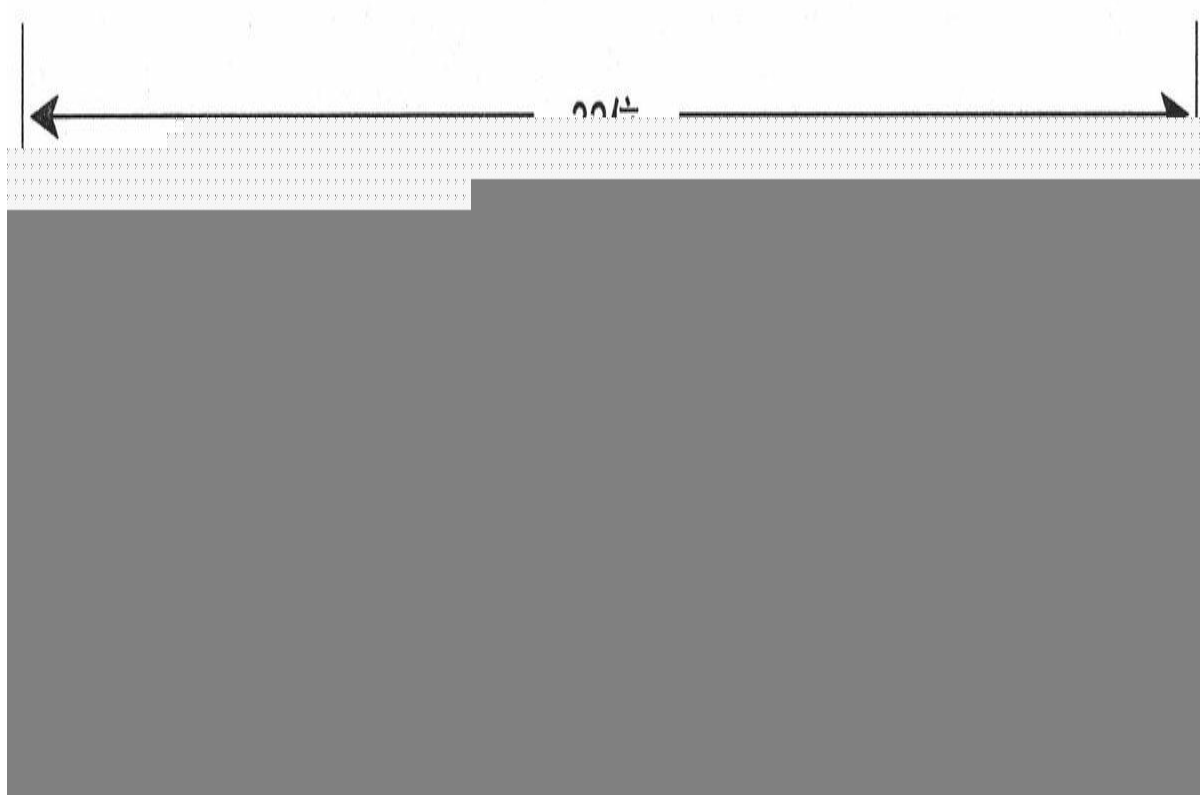


图7-10 路由器**Cayley**到达子网**10.1.7.0**的路由变成活动状态，并且为了确定可行后继路由器而去查询路由器**Lilienthal**

一旦从路由器**Lilienthal**收到一个答复消息，路由器**Cayley**就设置 $r=0$ ，路由也就变成了被动状态，如图7-12所示。路由器**Lilienthal**成为路由器**Cayley**的新后继路由器，FD也将设置成新的距离。最后，路由器**Cayley**发送给路由器**Lilienthal**一个包含其本地计算度量的更新。路由器**Lilienthal**也发送新的更新来通告它的新度量值。



图7-11 路由器**Lilienthal**有一个到达子网**10.1.7.0**的可行后继路由器。它执行一个本地计算，向路由器**Cayley**发送一个包含其经过路由器**Wright**到达目的子网的应答数据包，而且发送一个更新给路由器**Wright**

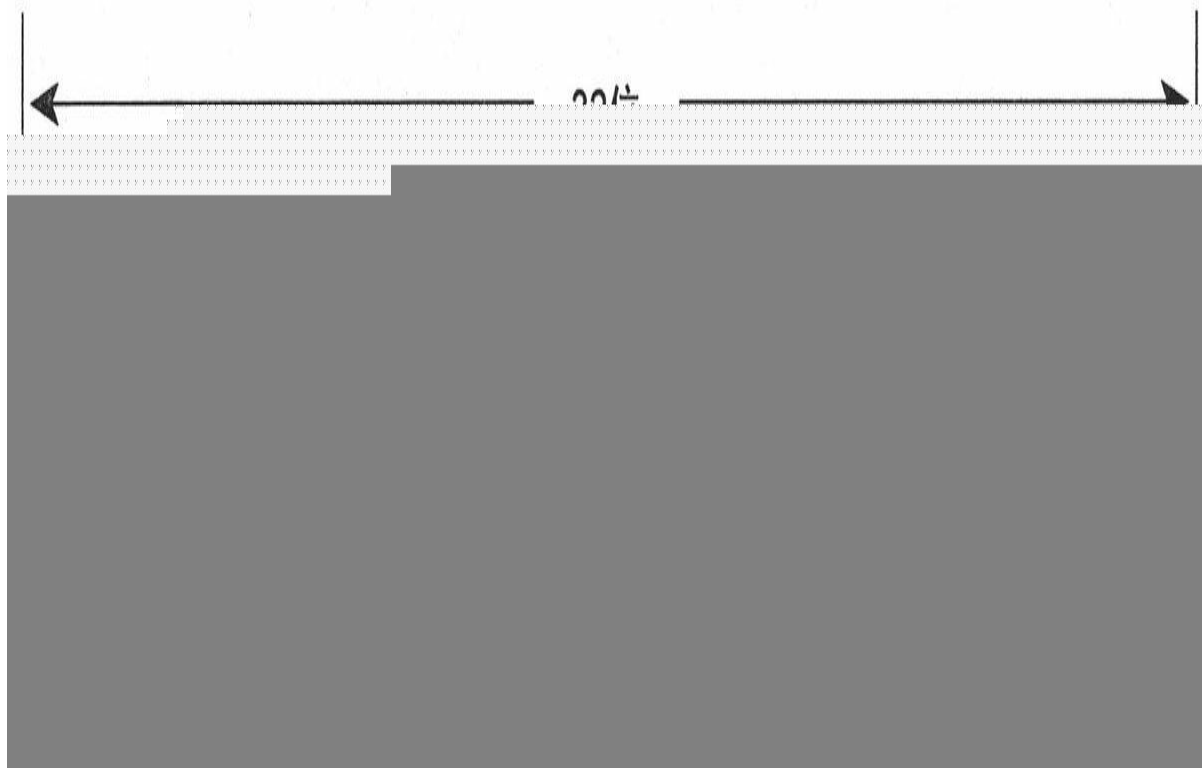
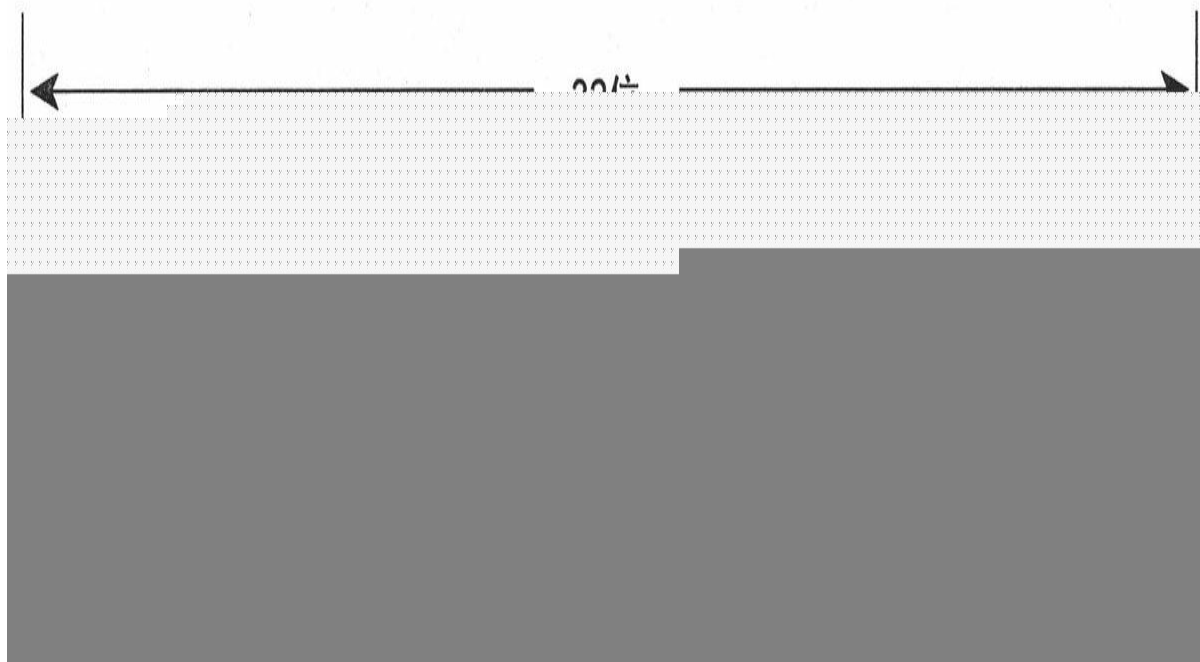


图7-12 路由器Cayley到达子网10.1.7.0的路由变为被动状态，而且发送一个更新给路由器Lilienthal

EIGRP协议包的行为可以通过调试命令**debug eigrp packets** 来观察。缺省情况下，路由器将会显示所有的EIGRP数据包。由于大量Hello和ACK调试信息的输出可能很难去跟踪，因此该命令允许使用关键字选项以便只显示指定的数据包类型。在示例7-13中，命令**debug eigrp packets query reply update** 用来在路由器Cayley上观察本例中描述的事件的数据包行为。

示例7-13 在本例中描述的**EIGRP**数据包事件可以通过这些调试信息进行观察



- **标记（Flags）** ——在输出的调试信息中，指出EIGRP包头中标记的状态；有关EIGRP包头的内容请参阅7.3.5小节。0x0表示没有标记被设置。0x1表示设置了初始化（initialization）位，在一个新的邻居关系中，当附加的路由条目是首个时，这个标记就被设置。0x2表示设置了条件接收位（conditional receive bit），这个标记用在私有的可靠组播算法中。
- **序列号（Seq）** ——表示一个数据包序列号/确认序列号。
- **idbq** ——表示在接口上的输入队列数据包数/输出队列数据包数。
- **iidbq** ——表示在接口上等待传送的不可靠组播数据包数/等待传送的可靠组播数据包数。
- **peerQ** ——表示在接口上等待传送的不可靠单播数据包数/等待传送的可靠单播数据包数。
- **serno** ——表示一个指向某条路由的双重连接的序列号的指针。这个指示器使用内部（或私有）机制，用来在一个快速变化的拓扑中跟踪正确的路由信息。

4. 扩散计算：范例2

这个例子所关注的是路由器Wright和它到达子网10.1.7.0的路由。虽然在这里描述的输入事件（在扩散更新计算的期间内链路的延迟变化了两次）在现实的网络中未必会发生，但是这个例子显示了DUAL算法是怎样控制多种度量变化的。

在图7-13中，把路由器Wright和Langley之间的链路代价由2变成10。经过路由器Langley到达10.1.7.0的距离现在超过了路由器Wright的可行距离，这将引起路由器Wright开始进行本地度量计算。这时度量将被更新，除了和链路代价发生改变的链路相连的邻居路由器Langley外，路由器Wright会向它所有的邻居发送更新消息，如图7-14所示。

请注意，路由器Langley是到达子网10.1.7.0的惟一可行后继路由器，这是因为路由器Chanute的本地计算度量高于路由器Wright的FD（ $1024 > 768$ ）。路由器Wright和Langley之间链路度量的增加引起路由器Wright在它的拓扑结构表中去查找一台新的后继路由器。由于路由器Wright在它的拓扑结构表中可以查到的惟一可行后继路由器是路由器Langley，因此，路由器Wright到达子网10.1.7.0的路由将变为活动状态。查询消息将被发送到邻居路由器，如图7-15所示。

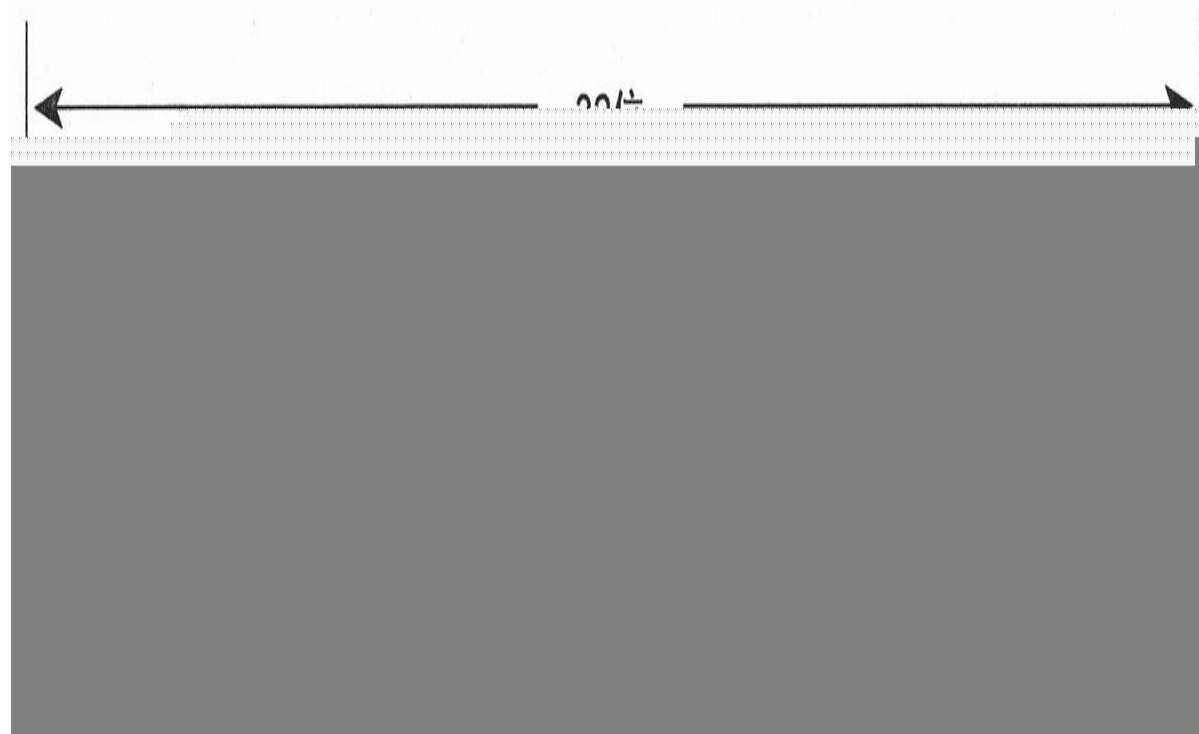


图7-13 路由器Cayley到达子网10.1.7.0的路由变为被动状态，并且向路由器Lilienthal发送一个更新消息

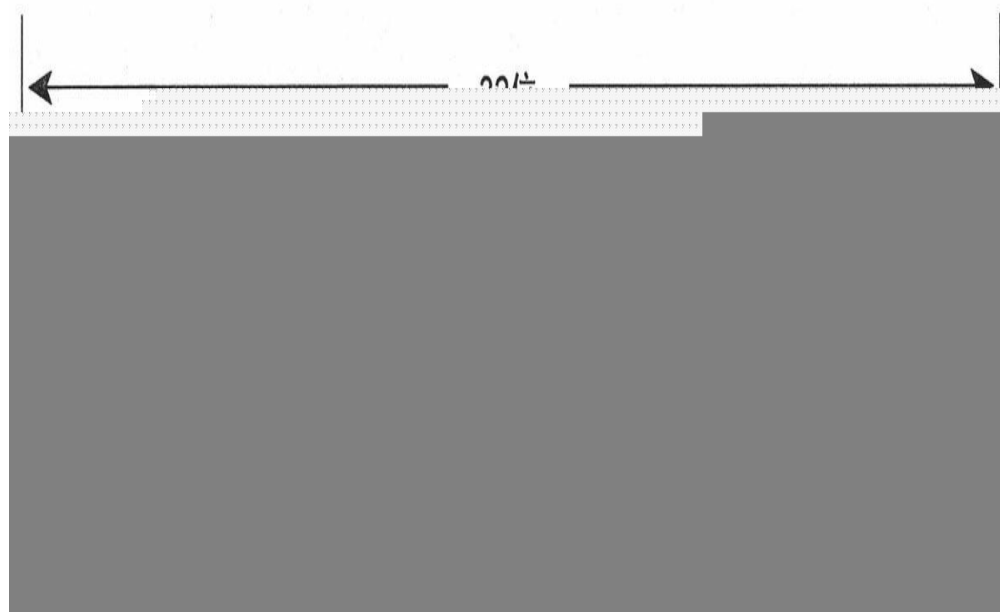


图7-14 路由器Wright发送含有新度量值的更新消息到除了路由器Langley之外的所有邻居路由器

同时，在图7-14中，路由器Wright发出的更新消息将引起路由器Cayley、Lilienthal和Chanute分别执行一个本地计算。

在路由器Cayley上，现在经过路由器Wright的路由距离超过了路由器Cayley的可行距离（FD）（ $2816 > 1024$ ）。因而，路由变为活动状态并向它的邻居路由器发送查询消息。

在图7-15中，路由器Lilienthal正在使用路由器Cayley作为一台后继路由器，但是还没有收到从路由器Cayley发出的查询消息。因此，路由器Lilienthal只不过重新计算了经过路由器Wright的路径的度量值，发现它不再满足可行性条件，从而从拓扑结构表中删除了这条路径。

在路由器Chanute看来，路由器Wright是它的后继路由器。因为路由器Wright所通告的距离不再满足路由器Chanute的可行性条件（FC）

（ $2816 > 1024$ ），并且因为路由器Chanute还有一台可行后继路由器（参见示例7-11），因此，路由器Chanute将会把路由器Wright从它的拓扑结构表中删除。随后，路由器Langley变成了路由器Chanute的后继路由

器，度量值被更新后，路由器Chanute向它的邻居路由器发送更新消息（参见图7-15）。而路由器Chaute上的路由则从来没有变为活动状态。



图7-15 路由器Wright到达10.1.7.0的路由变为活动状态，Wright向它的邻居发出查询，以便选择一个可行后继路由器。为了响应来自路由器Wright的早期更新，路由器Cayley使它的路由成为活动状态并向它的邻居路由器发出查询；同样，路由器Chanute也改变了它的度量值并发送出更新

路由器Cayley、Lilienthal和Chanute分别以不同的答复来响应发源于路由器Wright的查询，如图7-16所示。

路由器Cayley已经是活动状态，因为输入事件是来自于后继路由器的查询，这个查询最初标记为2（0=2），参见图7-7和表7-2。

路由器Lilienthal一旦收到路由器Wright的查询，就发送一个含有经过路由器Cayley的距离的应答。然而，就在刚发出这个应答消息后，路由器Lilienthal收到了来自路由器Cayley的查询，这时，路由器Lilienthal的可行距离超出了，因而度量值将被更新；路由器Lilienthal使路由变为活动状态，并向它的邻居路由器发出查询。

路由器Chanute已经把它的后继路由器切换到路由器Langley，并且只发出一个应答消息。

当上述的一切正在继续进行的时候，图7-16中显示了路由器Wright和Langley之间的链路代价又从10增加到了20。路由器Wright将又重新计算基于这个新的链路代价到达子网10.1.7.0的路由度量值，但是由于该路由此时是处于活动状态的，因而，在这条路由变成被动状态之前，它所通告的距离和可行距离（FD）都不能改变。

根据图7-7和表7-2，在路由处于活动状态的时候，到达目的地的距离如果增加了将会使 $0=0$ ，如图7-17所示。路由器Wright将响应来自于路由器Lilienthal的查询。路由器Wright所报告的距离还是它到达子网10.1.7.0的路由起初变为活动状态时的距离（记住，当路由处于活动状态时，路由器所通告的距离不能改变）。路由器Cayley也发送一个应答来响应路由器Lilienthal的查询。

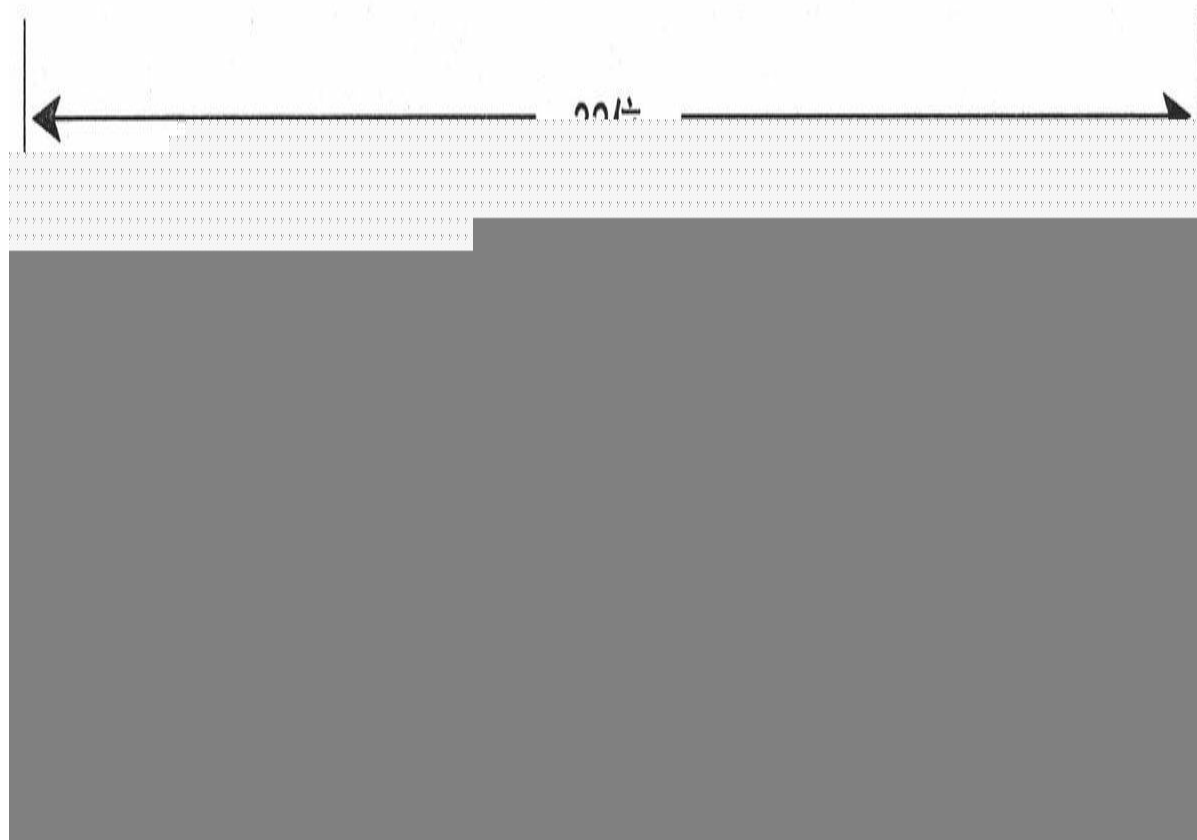


图7-16 路由器Cayley（a）答复来自路由器Wright的查询。路由器Lilienthal（b）答复来自路由器Wright的查询，并且（c）使路由变成

活动状态，同时发送查询来响应路由器**Cayley**的查询。路由器**Chanute**（d）回复路由器**Wright**的查询。路由器**Wright**（e）回复路由器**Cayley**的查询

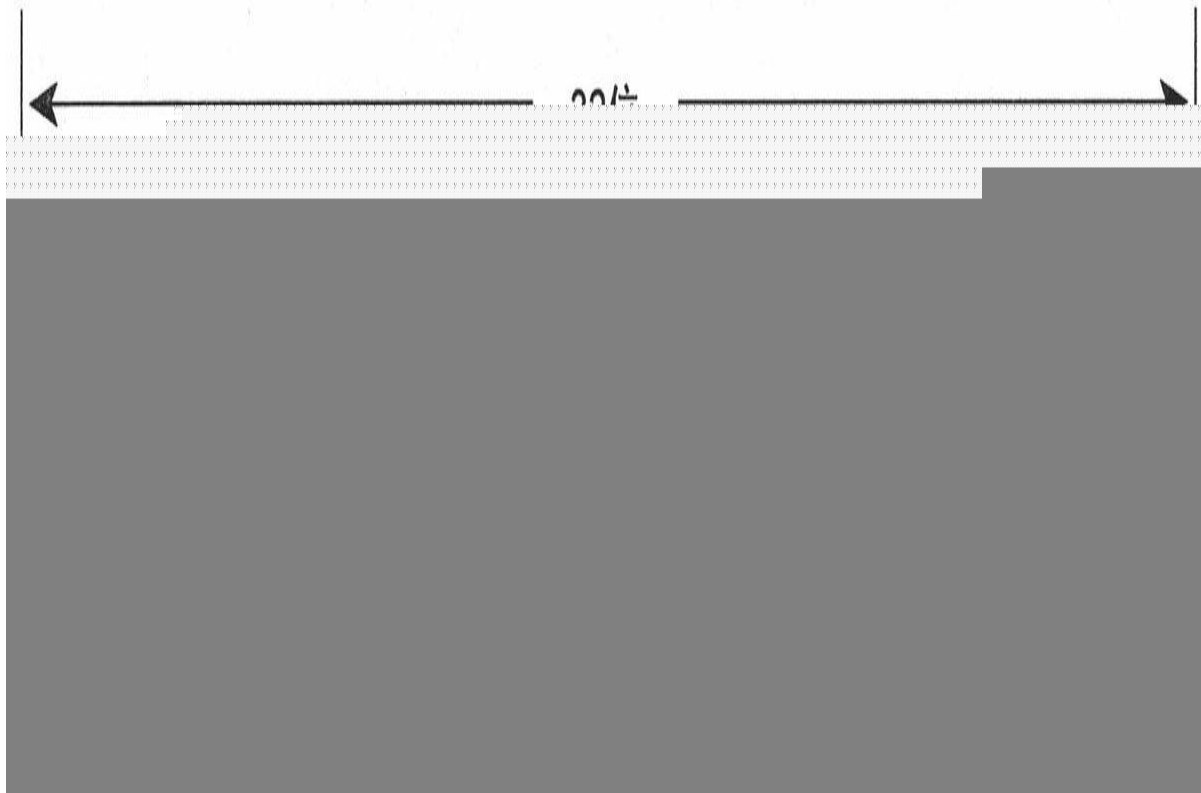


图7-17 在路由变为被动状态之前，路由器**Wright**不能改变它所通告的度量值

路由器**Lilienthal**在收到了它发送的所有查询的答复后，将把路由的状态转变成被动状态，如图7-18所示。这时，路由就可以设置新的可行距离（FD）了。由于路由器**Cayley**所通告的路由距离低于路由器**Lilienthal**的可行距离，因而它仍然保持是后继路由器。路由器 **Lilienthal**也发送一个答复来响应路由器**Cayley**的查询。



图7-18 收到了所希望得到的最后一个答复后，路由器Lilienthal将使它的路由转变为被动状态（ $r=0$ ， $o=1$ ）

图7-18中显示了路由器Wright和Langley之间的链路距离再次从20改变成15。路由器Wright也再次计算出它的路由的本地距离为4096，如图7-19所示。如果在路由变为被动状态之前，路由器Wright收到一个查询，那么它将仍然通告含有距离为2816的路由，2816是在路由变为活动状态时的距离值。

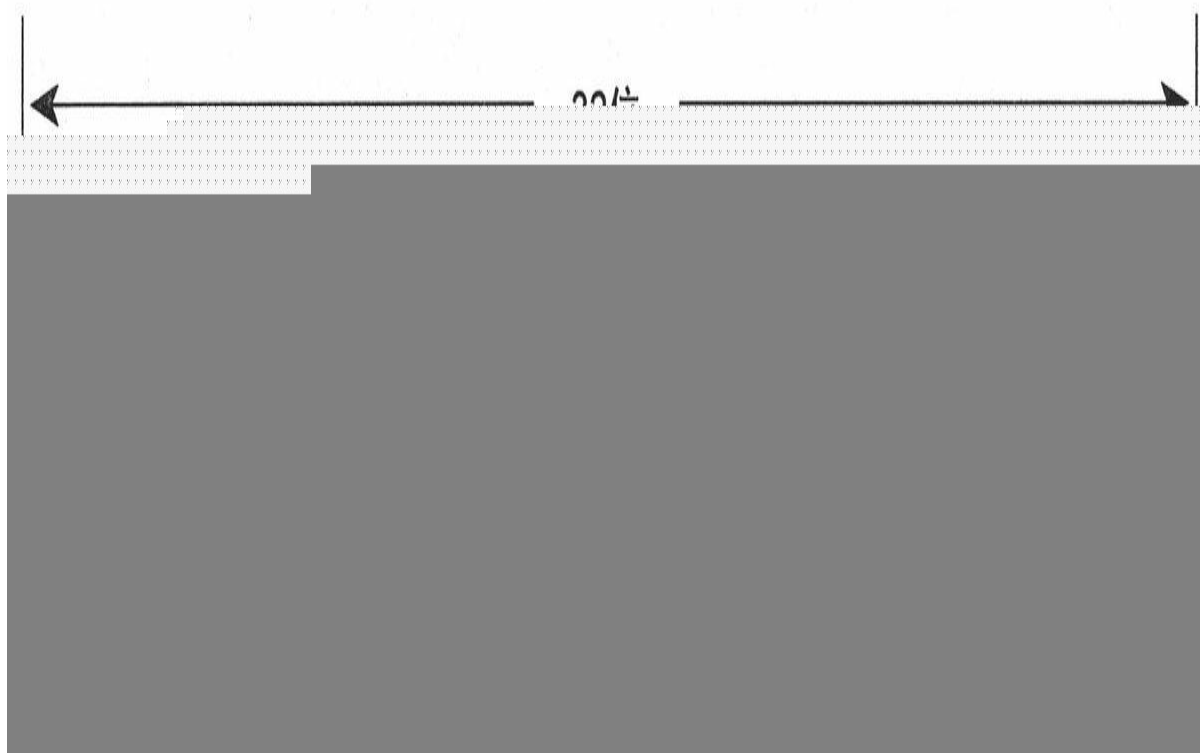


图7-19 收到了所希望得到的最后一个答复后，路由器**Cayley**将使它的路由状态转变为被动状态

当路由器**Cayley**收到它所发送的查询的答复时，它到达子网10.1.7.0的路由也将变成被动状态，如图7-19所示，将设置一个新的可行距离（FD）。虽然路由器**Wright**在本地计算的度量值是4096，但是它所通告的最新度量值却是2816。因此，路由器**Wright**满足路由器**Cayley**的可行性条件（FC），从而变为到达子网10.1.7.0的后继路由器，并发送一个答复给路由器**Wright**。

在图7-20中，路由器**Wright**收到了它所发出的每一个查询的答复后，它的路由状态就变为被动状态了。路由器**Wright**选择了路由器**Chanute**作为它的新后继路由器，并且把可行距离（FD）改变为路由器**Chanute**所通告的距离与它和邻居路由器（**Chanute**）之间的链路代价的总和。路由器**Wright**发送一个更新给它所有的邻居路由器，并通告它在本地所计算的新度量值。

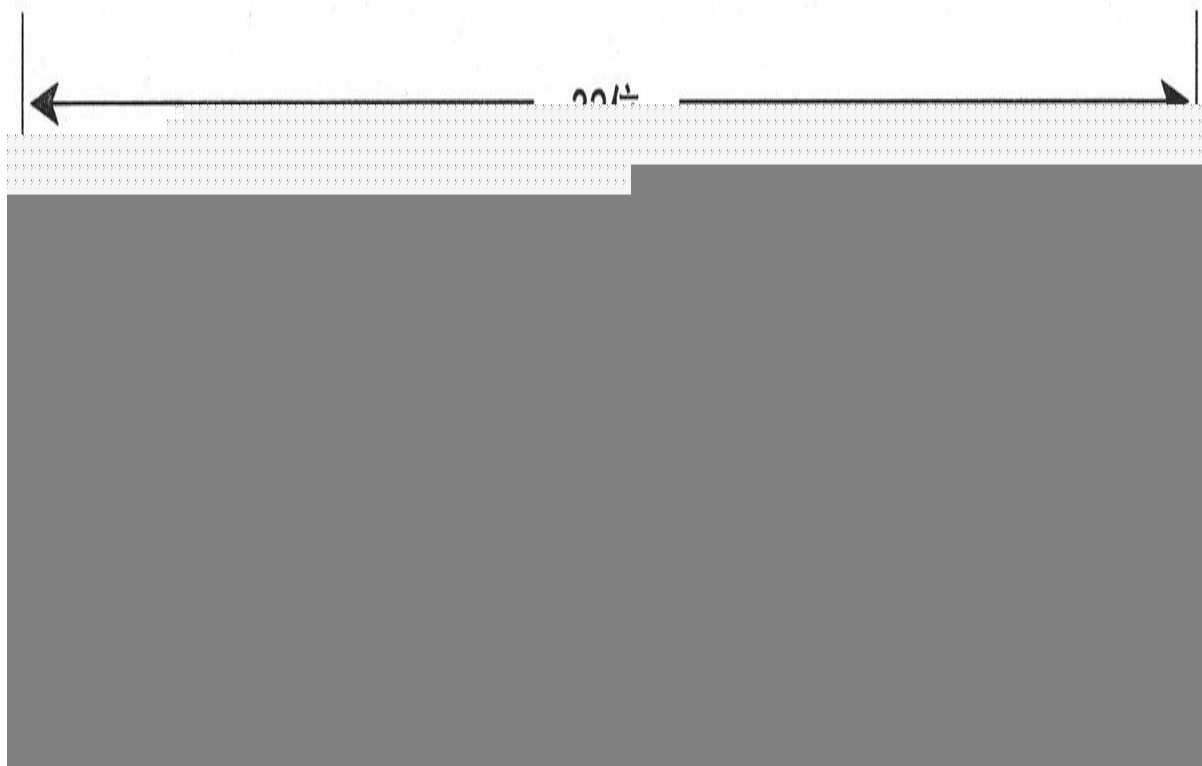


图7-20 路由器**Wright**转变到被动状态，选择路由器**Chanute**作为它的后继路由器，同时改变**FD**，并且更新所有的邻居路由器

路由器Cayley使用路由器Wright作为它的后继路由器。当它收到一个来自路由器Wright的并有较低代价的更新时，它将改变它在本地的计算度和**FD**，并且更新它的邻居路由器，如图7-21所示。

来自路由器Cayley的更新并不影响路由器Wright，这是因为路由器Cayley不满足路由器Wright的FC。在路由器Lilienthal上的更新会引起一个本地计算。

如图7-22所示，路由器Lilienthal减小了它的度量值及**FD**，并更新它的邻居。

虽然，扩散计算算法可以通过更加详细的描述或阅读一些读物以便完全的理解，但是在这里所讲述的内容和前面的例子也包含了扩散计算算法的主要核心内容：

- 任何时间，发生一个输入事件，就会执行一个本地计算。

- 如果在路由器的拓扑结构表中发现了一台或多台可行后继路由器，那么将使用具有最低度量代价的可行后继路由器作为它的后继路由器。

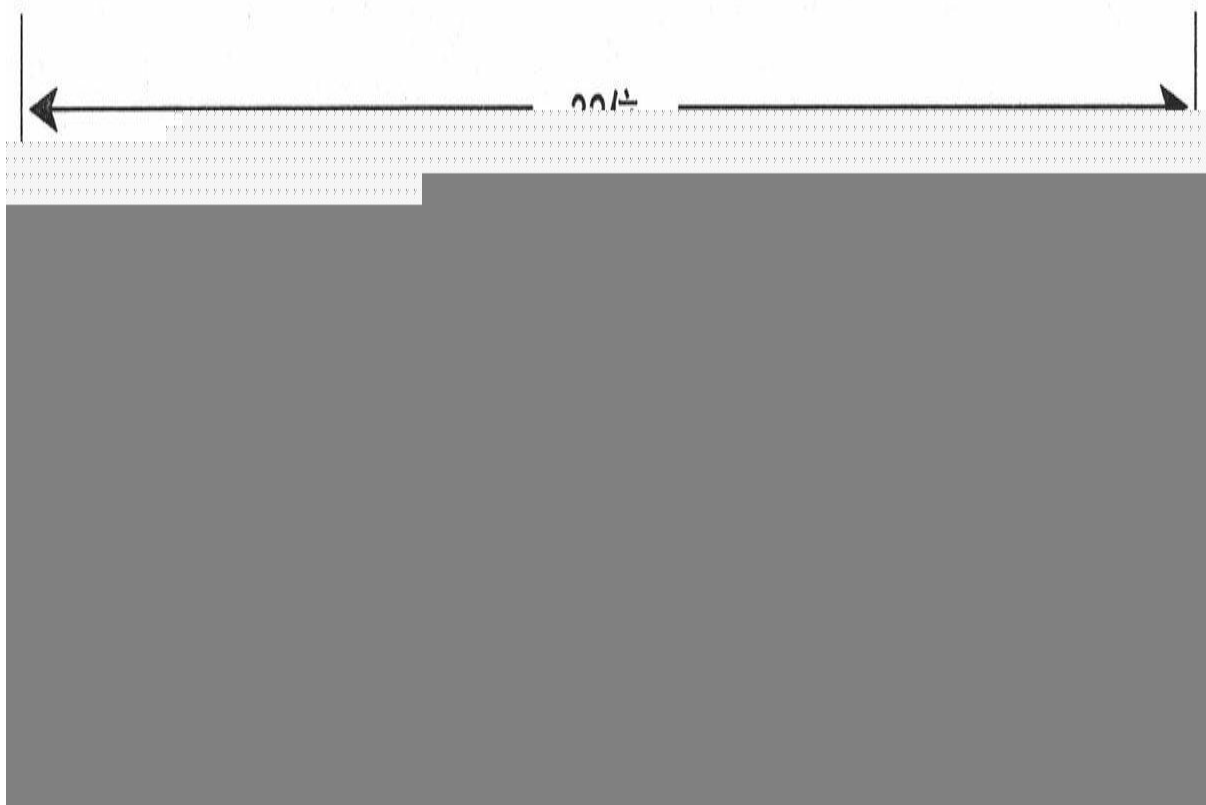


图7-21 路由器**Cayley**重新计算它的度量值，根据路由器**Wright**通告的较低的代价更改它的**FD**，并更新它的邻居路由器

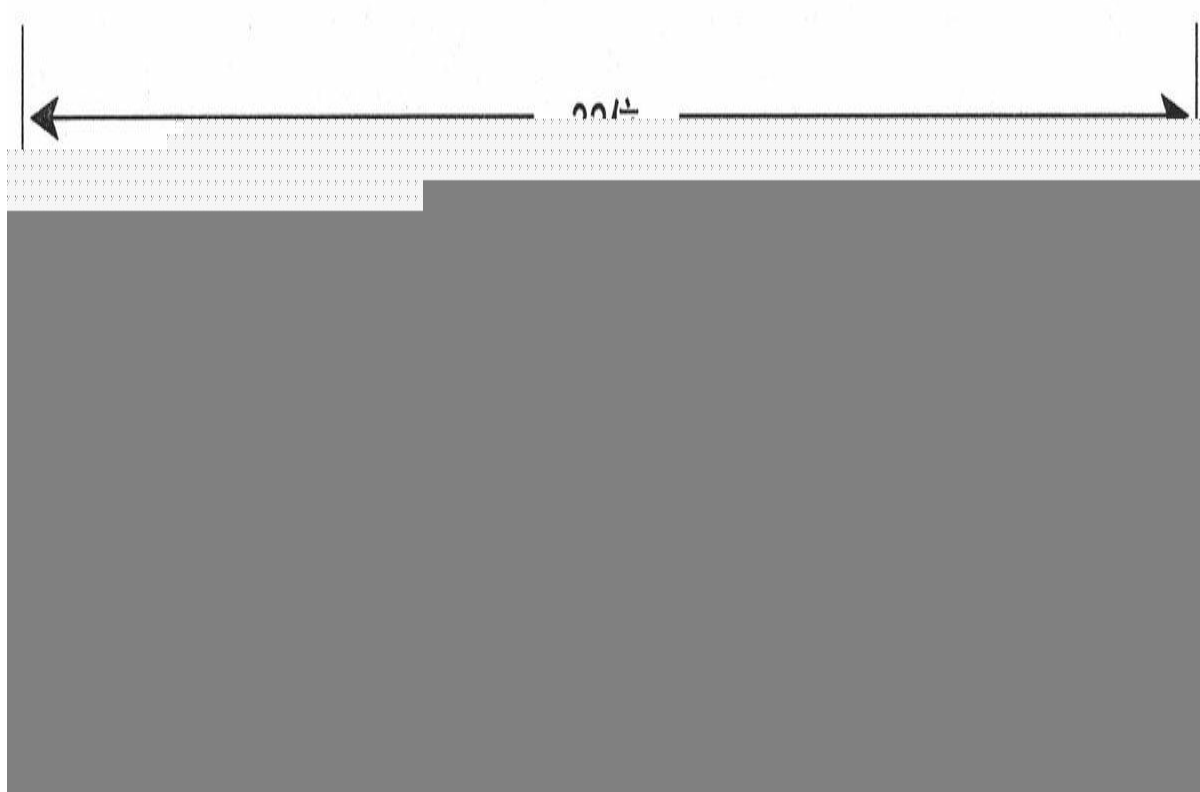


图7-22 路由器**Lilienthal**重新计算它的度量，基于来自路由器**Cayley**的更新改变它的**FD**，并更新它的邻居路由器

- 如果没有发现可行后继路由器，那么将使它的路由变成活动状态，向它的邻居路由器发送查询消息，以便确定一个可行后继路由器。
- 在所有的查询被答复响应之前，或者活动计时器计时超时之前，将保持路由状态为活动状态。
- 如果扩散计算的结果无法发现一个可行后继路由器，那么将宣告这个目的地不可到达。

7.3.5 EIGRP的数据包格式

EIGRP协议数据包的IP头部指定它的协议号是88，最大长度可以是传输该数据包接口的IP最大传输单元（MTU）的大小——通常是1500字节。紧接着IP头部后面的是EIGRP协议头部，EIGRP协议头部后面是类型/长度/数值（Type/Length/Value, TLV）这3个参数的不同组合。这些TLV不

仅携带路由条目的信息，而且提供多个字段来管理DUAL算法的处理、组播的先后次序和IOS软件版本。

1. EIGRP包头

图7-23中显示了EIGRP的头部，它是每个EIGRP数据包的开始部分。

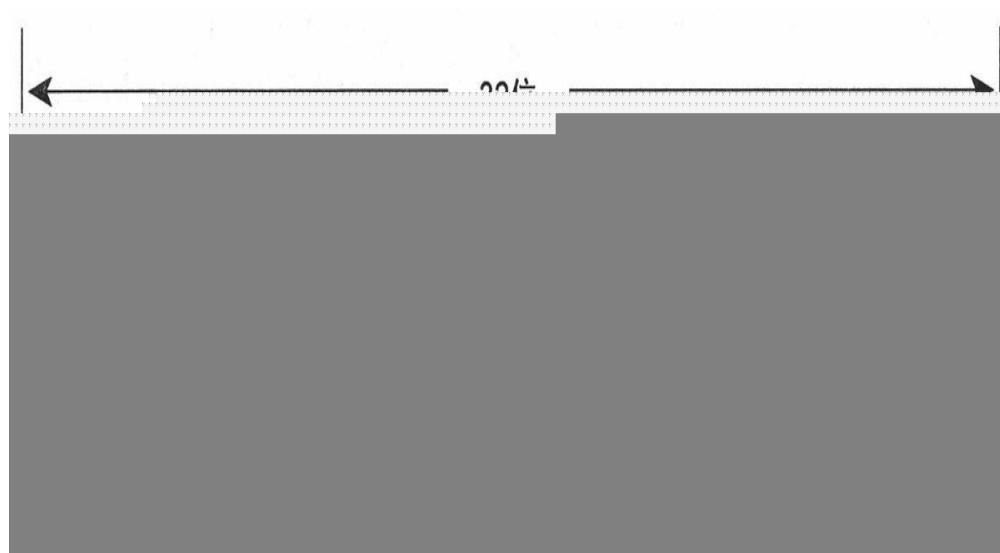


图7-23 EIGRP包头的格式

- **版本号（Version）**——指出发起EIGRP进程的具体版本。EIGRP协议本身的版本自发布后还没有改变过。
- **操作码（Opcode）**——指出EIGRP数据包的类型，这显示在表7-3中。虽然表中包含了IPX SAP数据包类型，但IPX EIGRP的讨论已经超出了本书的范围。

表7-3 EIGRP的数据包类型

操作码 (Opcode)	类型 (Type)
1	更新 (Update)
3	查询 (Query)
4	答复 (Reply)
5	问候 (Hello)
6	IPX SAP

SIA查询

SIA答复

- ### 表7-4 类型/长度/数值 (TLV) 的类型

数值	TLV类型
一般的 TLV 类型	
0x0001	EIGRP参数
0x0003	序列（Sequence）
0x0004	软件版本*
0x0005	下一个组播序列
IP 特有的 TLV 类型	

0x0102	IP内部路由
0x0103	IP外部路由
AppleTalk 特有的TLV 类型	
0x0202	AppleTalk内部路由
0x0203	AppleTalk外部路由
0x0204	Apple Talk电缆配置
IPX 特有的TLV 类型	
0x0302	IPX内部路由
0x0303	IPX外部路由

* 这个数据包指出是软件的老版本在运行（软件版本为0）还是软件从IOS10.3（11）、11.0（8）和11.1（3）起的新版本（软件版本为1）在运行。

2. 一般的TLV字段

这些TLV字段可以携带EIGRP的管理信息而不需要指定任何一个可路由的协议。带参数的TLV用来传递度量权重和抑制时间，如图7-24所示。序列、软件版本和下一个组播序列等TLV是用于Cisco的私有可靠性组播算法的，该内容已超出了本书的讲述范围。

3. IP特有的TLV字段

每一个内部路由和外部路由的TLV都包含一个路由条目。每个更新、查询和答复数据包都至少包含一个路由TLV。

内部路由和外部路由的TLV包括了路由的度量信息。就像早先提到的，EIGRP协议使用的度量与IGRP协议使用的度量相同，只是扩大了256倍。



图7-24 带EIGRP参数的TLV

（1）IP内部路由的TLV

内部路由是指在EIGRP自主系统内部可以到达目的地的路径。内部路由的TLV格式如图7-25所示。



图7-25 IP内部路由TLV

- 下一跳（**Next Hop**）——是指下一跳IP地址。这个地址可能是、也可能不是始发路由器的地址。
- 延迟（**Delay**）——是指所配置的以10μs为单位表示的延迟总和。注意，不像IGRP数据包中24位的字段，这个字段是32位的。这个更大的字段可以容纳EIGRP使用的大256倍的延迟。一个0xFFFFFFFF的延迟标识一个不可到达的路由。
- 带宽（**Bandwidth**）——就是 $256 \times BW_{IGRP (min)}$ ，或者用

2560000000除以沿着路由方向的所有接口所配置的最小带宽。像延迟一样，这个字段也比IGRP的带宽字段多8位。

- **MTU** ——是指沿着到达目的地的路由上所有链路中最小的最大传输单元。虽然EIGRP数据包中包含了这个参数，但是它从来没有在度量值的计算中使用过。

- **跳数（Hop Count）** ——是一个在0x01~0xFF之间的数字，表示到达目的地的路由的跳数。路由器将通告与之直连网络的跳数为0跳；后续的路由器将记录并通告相对于下一跳路由器的路由。

- **可靠性（Reliability）** ——是一个在0x01~0xFF之间的数字，用来反映沿着到达目的地的路由上接口的出站误码率的总和，每5min通过一个指数的加权平均来计算。0xFF表示100%的可靠链路。

- **负载（Load）** ——是一个在0x01~0xFF之间的数字，用来反映沿着到达目的地的路由上接口的出站负载的总和，每5min通过一个指数的加权平均来计算。0x01表示一条最小负载的链路。

- **保留字段（Reserved）** ——一个未使用的字段并且总是设置为0x0000。

- **前缀长度（Prefix Length）** ——指出一个地址掩码中的网络位的个数。

- **目的地址（Destination）** ——表示一个路由的目的地址。虽然在图7-25和图7-26中显示的字段只是一个3个八位组字节长的字段，但是这个字段针对不同的地址是可变的。例如，假如有一条到达目的地址10.1.0.0/16的路由，它的前缀长度是16，因而目的地址只需要一个包含10.1的2个八位组字节的字段。假如一条到达目的地址192.168.17.64/27的路由，它的前缀长度是27，因而目的地址将需要一个包含192.168.17.64的4个八位组字节的字段。如果这个字段没有3个八位组字节的长度，那么TLV将增加0来填充这个字段，以便使这个字段达到4个八位组字节的边界长。

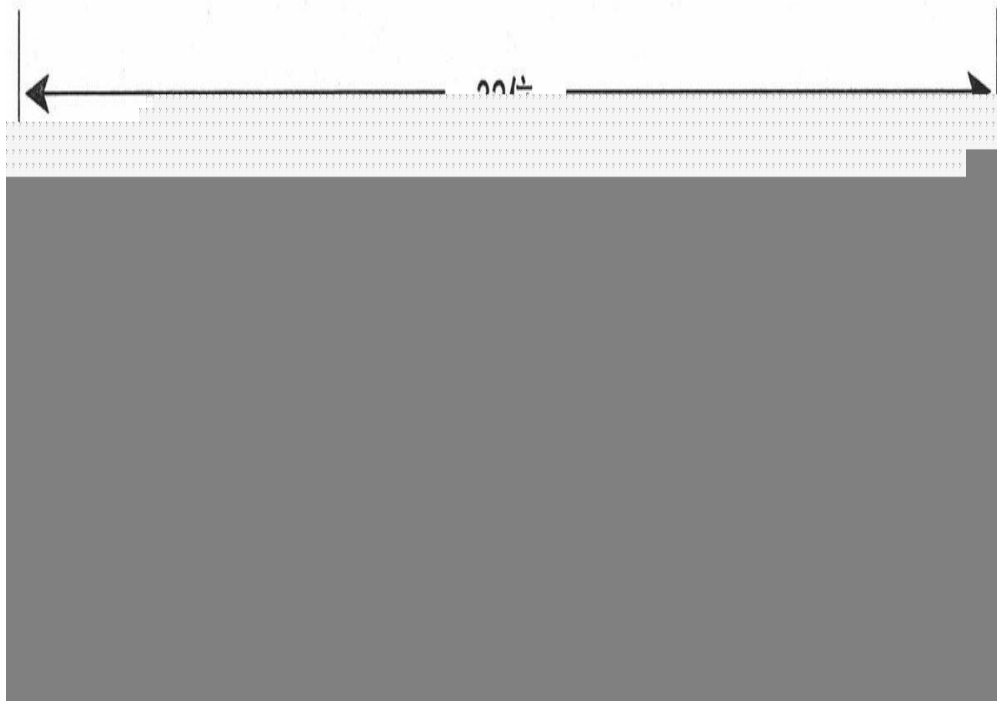


图7-26 IP外部路由的TLV

（2）IP外部路由的TLV

外部路由是指到达EIGRP自主系统外部的目的地址的一条路径，或者是一条通过路由重新分配注入到EIGRP域内的路由。图7-26显示了外部路由TLV字段的格式。

- 下一跳（**Next Hop**）——就是路由的下一跳IP地址。在一个多路访问的网络中，正在通告路由的路由器可能不是到达目的地的最佳下一跳路由器。例如，一个在以太网链路上宣告EIGRP路由的路由器也可能宣告BGP的路由，同时也可能把从BGP学到的路由通告到EIGRP的自主系统。因为以太网链路上的其他路由器并不宣告BGP路由，因此，它们也无法得知BGP宣告者的接口是一个最佳的下一跳地址。下一跳 字段允许同时宣告两种路由协议的路由器告诉它的EIGRP邻居——“使用地址A.B.C.D代替我的接口地址作为它的下一跳”。
- 原路由器（**Originating Router**）——是一个IP地址，或者重分配外部路由到EIGRP自主系统的路由器ID。

- 原自主系统号（**Originating Autonomous System Number**）——是指始发路由的路由器所在的自主系统号。
- **Arbitrary Tag** ——可以用来携带一组路由映射的标记。如要了解路由映射的用法，请参见第14章的内容。
- 外部协议度量（**External Protocol Metric**）——顾名思义，这是一个外部协议的度量。在和IGRP协议之间进行重分配时，这个字段用来跟踪IGRP协议的度量值。
- 保留字段（**Reserved**）——一个未使用的字段并且总是设置为0x0000。
- 外部协议ID（**External Protocol ID**）——用来标识外部路由是从哪一个协议学习到的。表7-5列出了这个字段的可能值。

表7-5

外部协议ID字段的数值

表7-5

外部协议ID字段的数值

代码	外部协议
0x01	IGRP
0x02	EIGRP
0x03	静态路由
0x04	RIP
0x05	Hello
0x06	OSPF
0x07	IS-IS
0x08	EGP
0x09	BGP
0x0A	IDRP
0x0B	直连链路

- 标记（**Flags**）——目前仅定了两个标记。如果这个8位字段最右边的第一位设置了（0x01），该路由就是外部路由。如果右边的第二位设置了（0x02），该路由就是一个候选的缺省路由。缺省路由

的讲述请参见第12章的内容。

其余的字段描述了度量和目的地址。这些字段的含义与在内部路由TLV中讲述的相同字段的含义是一样的。

7.3.6 地址聚合

在第1章中介绍了子网划分的操作方法——为了使多条链路可以使用一个主网络地址，将地址掩码扩展到了主机地址空间。第6章介绍了可变子网掩码的操作方法——地址掩码的使用扩展到了子网中，甚至在子网中又创建了更多的新子网。

从相反的观点来看，子网地址也可以考虑成为一组更小的子网的汇总，而一个主网络地址可以看作一组子网的汇总。在每个这样的实例中，汇总都是通过减少子网掩码的长度来完成的。

地址聚合是打破主网络地址分类限制的进一步汇总措施。聚合的地址表示了一组数字上连续的网络地址，或称为超网（supemet）。[\[15\]](#)图7-27中显示了一个聚合地址的例子。



图7-27 这组网络地址可以看作是单个的聚合地址或聚合子网

图7-28显示了是怎样得出图7-27中的聚合地址的。对于一组网络地址，寻找出所有网络地址的共同位并对这些位进行掩码。被掩码覆盖的部分就是聚合地址。

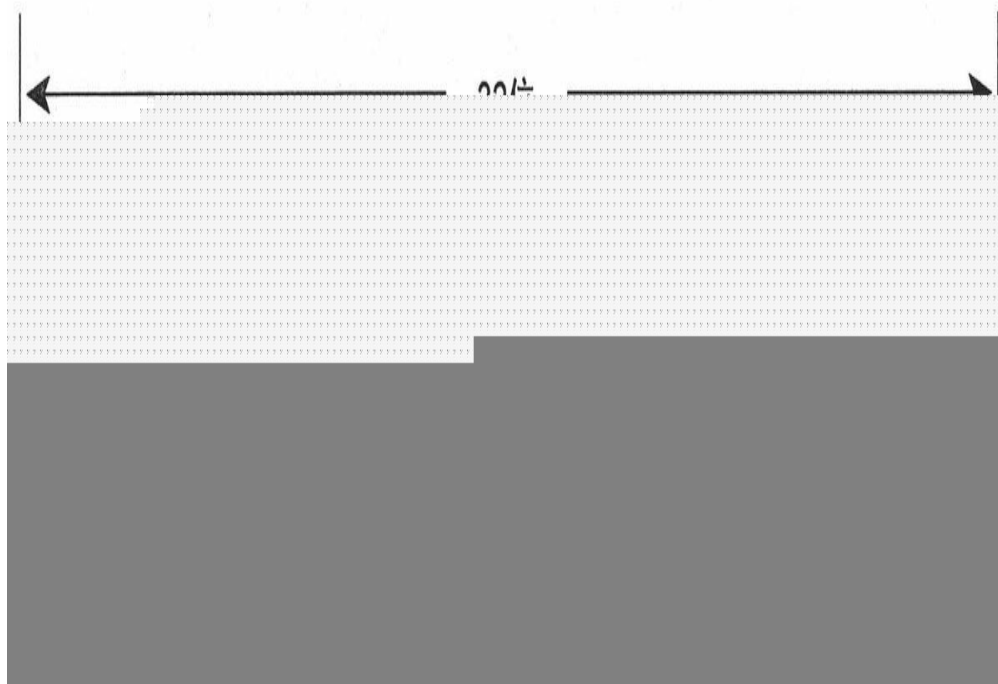


图7-28 聚合地址是由对一组数字上连续的网络地址的所有共同位进行掩码而得出的

当设计一个超网时，有一点很重要，就是超网的成员地址应该由原来掩码位的一个完整和连续的地址集合组成。例如，在图7-28中，聚合地址的20位掩码比成员地址的掩码少4位。对于这4个“不同”的位，注意，它们包括了0000~FFFF之间二进制位组合的每一种可能性。按照这个设计规则如果失败的话，将会引起寻址方案冲突、减小聚合路由的性能，并且可能导致路由选择环路和路由选择“黑洞”。

汇总寻址的一个明显的好处就是节省网络资源。由于通告更少路由从而节省了带宽，而处理更少路由则节省了CPU的周期。更为重要的是，由于减小了路由表的大小而节省了内存的使用。

无类别路由选择、VLSM和聚合寻址都是通过创建层次化的地址来达到最大限度地节省网络资源的。与IGRP协议不同，EIGRP协议支持所有这些寻址策略。在图7-29中，Treetop Aviation的工程部门分配了16个C类的地址，这些地址已经根据需要分配到不同的子部门中。

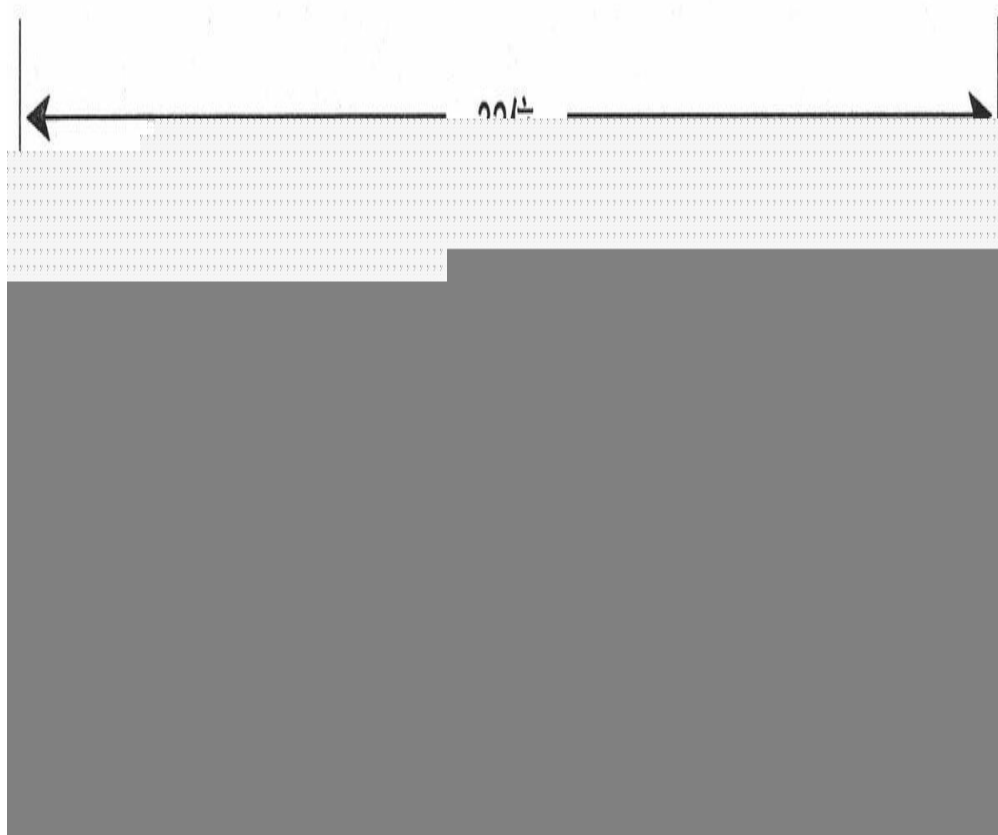


图7-29 在**Treetop Aviation**中，一个更大部门中的几个子部门正在聚合地址。依次地，整个部门将通过单个聚合地址**192.168.16.0/20**通告出去

发动机、电力和水力部门的聚合地址是它们自己聚合到单个地址**192.168.16.0/21**中去的。这个地址和机身部门的聚合地址一起被聚合到一个单一的地址**192.168.16.0/20**中，这个地址也表示了整个工程部门的地址。

其他的部门也可以进行类似地表示。例如，假设**Treetop Aviation**总共有8个部门，而每个部门分配的地址与工程部门的相似，那么在最高层次的骨干路由器只有很少的8条路由，如图7-30所示。

层次化的地址设计将继续应用于每个部门的每一个子部门中，通过子网化划分成更小的单独的网络地址；**VLSM**也可以用来进一步划分子网。路由选择协议将在网络的边界上自动地汇总子网，这和前面章节的讲述是一致的。

在Internet上，地址聚合也允许地址的节省和地址的分层。对于以指数速度增长的Internet，有两点需要关注：可供使用的IP地址（尤其是B类地址）的消耗和存储Internet路由选择信息所需要的巨大的数据库。

这个问题的一种解决方案就是使用称为无类别域间路由选择（CIDR）的方法。[\[16\]](#)在CIDR下，C类地址的聚合由IANA机构分配给国际上不同的地址分配权威机构，像亚太地区的亚太网络信息中心（Asia Pacific Network Information Centre, APNIC），北美地区的美洲Internet编号注册局（American Registry for Internet Numbers, ARIN），以及欧洲地区的Rèseaux IP Européens（RIPE）机构。这些地址聚合是按照地域进行分配的，如表7-6所示。

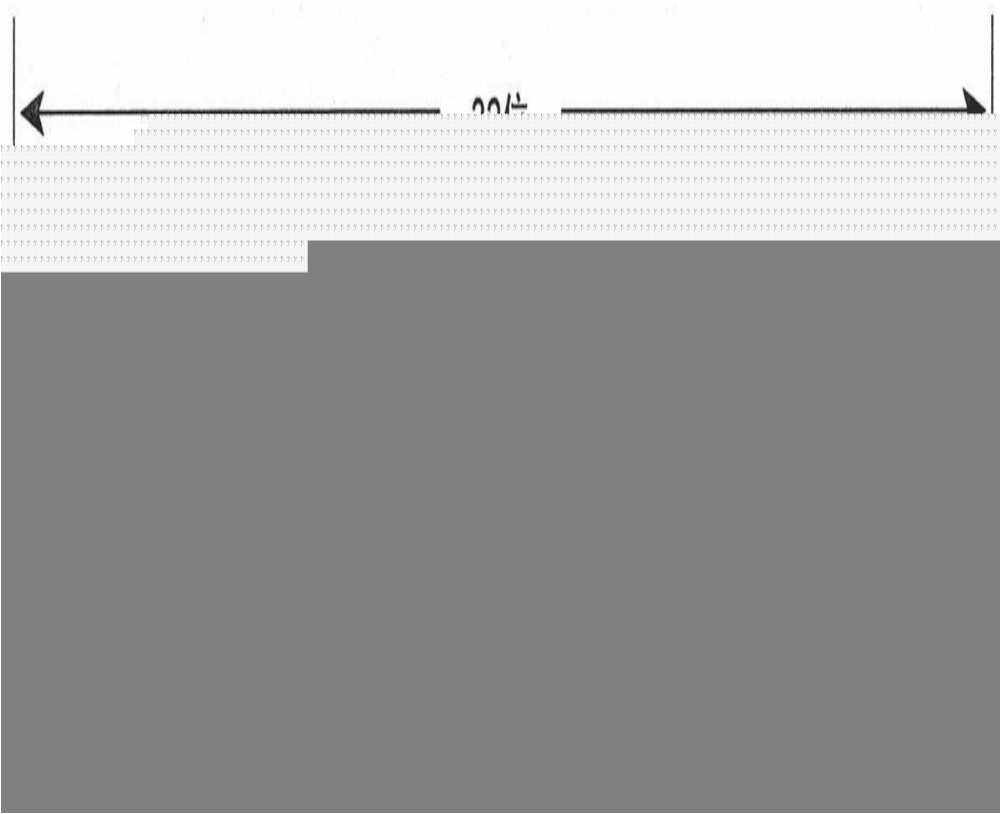


图7-30 虽然在这个网络中有**128**个主网络地址并且可能覆盖了**32000**台主机，但是骨干路由器的路由表中只有**8**条聚合地址

表7-6 **CIDR地址按国际上的地理区域分配**

地区	地址范围
----	------

Multiregional	192.0.0.0-193.255.255.255
欧洲	194.0.0.0-195.255.255.255
其他	196.0.0.0-197.255.255.255
北美	198.0.0.0-199.255.255.255
中、南美	200.0.0.0—201.255.255.255
环太平洋地区（Pacific Rim）	202.0.0.0-203.255.255.255
其他	204.0.0.0-205.255.255.255
其他	206.0.0.0-207.255.255.255

这些地址分配权威机构轮流地把他们自己管理的那部分地址分配给本地的网络服务提供商（ISP）。当一个组织申请IP地址并且所需的地址小于32个子网和4096台主机时，将可以分配给它一组连续的称为CIDR块（CIDR block）的C类地址。

这样，各个组织团体所属的Internet路由器就可以把单一的汇总地址通告给他们的ISP。反过来，ISP也可以把其自己所有的地址聚合起来，因此在理论上可以把世界上一个区域内的所有ISP的地址都汇总到表7-6中所表示的地址中去。目前所了解到的Internet全球路由表的大小接近200000条路由，显然并没有很好的坚持使用CIDR技术。尽管如此，CIDR技术也是一个很好的理念。在规范IPv6地址分配给地区管理机构方面也进行了类似的努力。我们只能希望在IPv6方面的努力能够比在IPv4地址分配方面更加成功。

7.3.7 EIGRP和IPv6

正当撰写本书第二版时，EIGRP协议还不支持IPv6协议。但是，对EIGRP协议进行扩展从而支持IPv6协议的努力也在进行，也许在读者阅读到本书的时候将会实现这些扩展。由于EIGRP数据包使用TLV传送数据，因此扩展该协议支持IPv6将可以通过增加专门的IPv6 TLV来简单的实现。

7.4 配置EIGRP

本节的案例研究演示了一个基本的EIGRP配置，并讲述了路由汇总的技巧和与IGRP协议之间的互操作性。

7.4.1 案例研究：EIGRP的基本配置

EIGRP只需要两个步骤就可以启动一个EIGRP的路由选择进程：

步骤1： 使用**router eigrp process-id** 命令启动EIGRP进程。

步骤2： 使用**network** 命令来指定运行EIGRP协议的每个主网络。

EIGRP进程ID号可以是1~65535（0不允许使用）之间的任何一个数字，只要对必须共享路由信息的所有路由器上的所有EIGRP进程ID号是相同的，那么网络管理员可以随意地选用进程ID号。另外，这个进程号也可以是公共分配的自主系统号。图7-31显示了一个简单的网络，图中3台路由器的配置参见示例7-14~示例7-16。

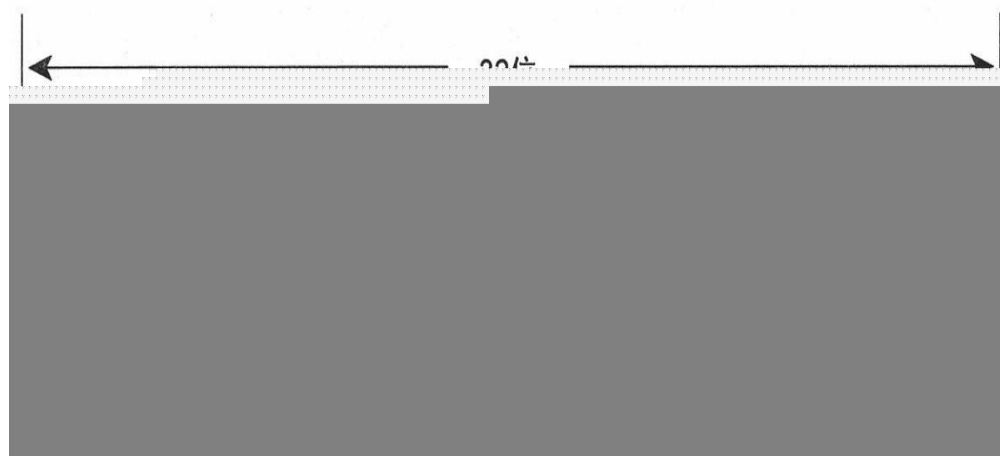


图7-31 与IGRP协议不同，EIGRP协议可以支持这个网络的VLSM要求

示例7-14 路由器Earhart的配置



示例7-15 路由器**Cochran**的配置



示例7-16 路由器**Lindbergh**的配置



路由器Earhart的路由表参见示例7-17。这个路由表显示了EIGRP协议缺省的管理距离是90，并且表中的网络172.20.0.0被划分为不同的子网。

示例7-17 路由器**Earhart**的路由表



与本章前面的一些例子不同，图7-31中的网络使用了缺省度量，这样在一个更为实际的环境中复习一下EIGRP的度量计算也许是有用的。

跟踪从路由器Earhart到达网络192.168.16.0的路由，这条路由的路径穿过了一个串行接口和一个以太网接口，每个接口的度量都是缺省的配置数值。这条路由路径的最小带宽是串行接口上的带宽，[\[17\]](#)而时延是这两

个接口时延的总和。请参考表7-1：

$$BW_{\text{EIGRP (min)}} = 256 \times 6476 = 1\,657\,856$$

$$DLY_{\text{EIGRP (sum)}} = 256 \times (2000 + 100) = 537\,600$$

因此，

$$\text{Metric} = 1\,657\,856 + 537\,600 = 2\,195\,456$$

7.4.2 案例研究：非等价负载均衡

在和RIP协议同样的CEF/快速交换/处理交换（CEF/fast/process switching）转发机制的限制下，EIGRP可以在最多16条等价的路由路径[18]上实现等价负载均衡。但与RIP协议不同的是，EIGRP协议也可以实现非等价的负载均衡。在图7-32中，路由器Earhart和路由器Cochran中间另外增加了一条串行接口，它的带宽配置为256kbit/s。这样做的目的是，为了使路由器Earhart在这两条链路上可以执行非等价负载均衡，即根据它们链路度量大小的反比来分配这两条链路上数据流量的负载大小。

观察路由器Earhart通过S0/1接口到达网络192.168.17.0的路由，可以看出最小的带宽是1 544 kbit/s（假定路由器Cochran的以太网接口使用的是快速以太网的缺省带宽，即 100 Mbit/s）。根据表7-1中所标明的，串行接口和快速以太网接口的 $DLY_{\text{EIGRP (sum)}}$ 为 $256 \times (2000 + 10) = 514560$ 。

$BW_{\text{EIGRP (min)}}$ 为 $256 \times (10^7 / 1544) = 1657856$ ，因此，这条路由的复合度量值是 $514560 + 1657856 = 2172416$ 。

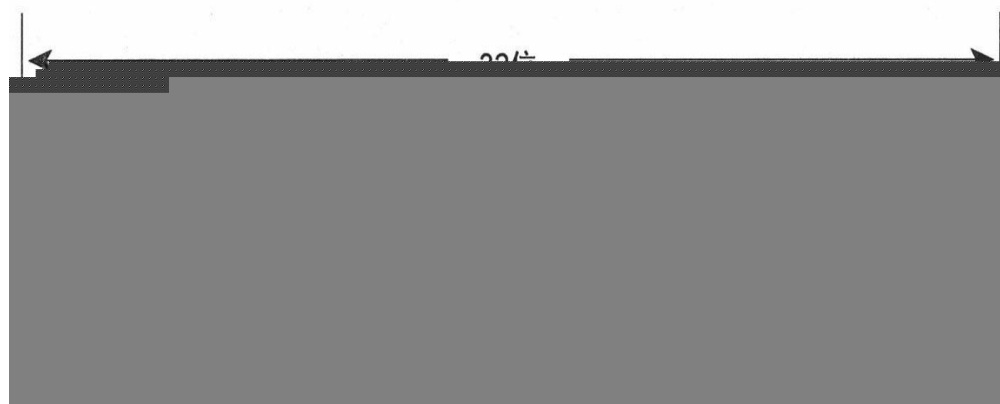
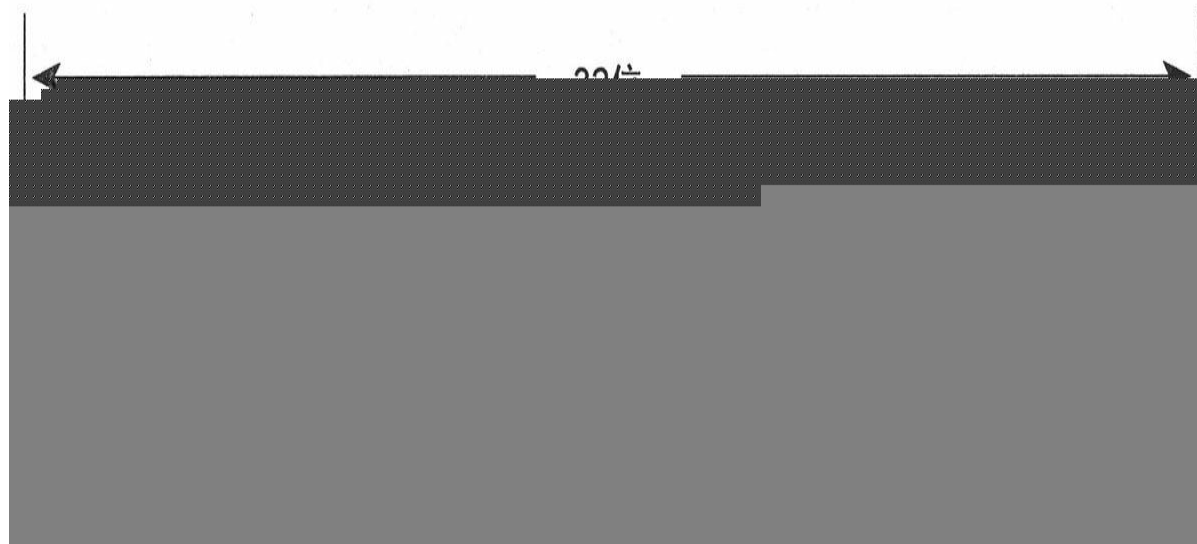


图7-32 可以配置**EIGRP**协议在链路之间实现非等价负载均衡，例如路由器**Earhart**和**Cochran**之间的链路

路由器Earhart通过S0.3接口到达网络192.168.17.0的最小带宽是256kbit/s。它的 $DLY_{EIGRP}(\text{sum})$ 与上面的第一条路由相同，因此，这条路由的复合度量值是 $256 \times (10^7 / 256) + 514560 = 10514432$ 。如果不做进一步的配置，**EIGRP**协议将会简单地选择路径代价最小的路径。示例7-18显示了路由器Earhart仅仅使用经过串行接口S0.1的那条路径，它的度量值为2172416。

示例7-18 路由器**Earhart**仅仅使用最小路径代价的链路到达网络**192.168.17.0**。要实现非等价负载均衡则需要另外的配置



差异变量（**variance**）命令用来确定哪些路由在非等价负载均衡中是可以使用的。**Variance**定义了一个倍数因子，用来表示一条路由的度量值和最小代价路由的差异程度。任何路由的度量值如果超过了最小代价路由的度量值乘以**Variance**的值，那么这条路由将不被使用。

Variance的缺省值是1，这意味着如果要想实现负载均衡，多条路由的度量值必须是相同的。**Variance**必须是整数。

路由器Earhart通过S0.3接口的路由的度量值是通过S0.1接口的路由的 $10514432 / 2172416 = 4.8$ 倍。因此，为了在这两条链路上实现非等价的负载均衡，路由器Earhart上的**variance**值应该是5。**EIGRP**的配置参见示例7-19。

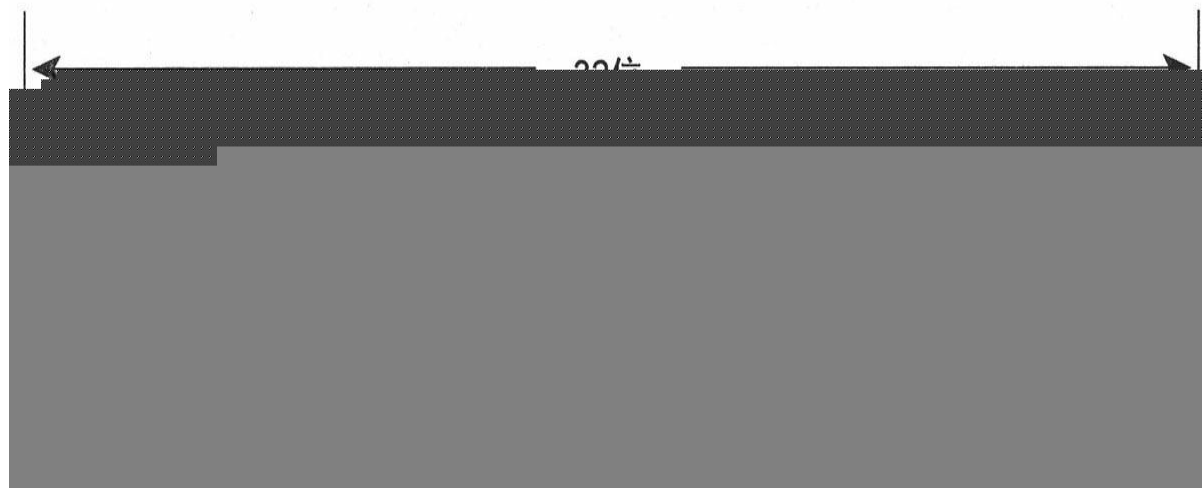
示例7-19 在路由器**Earhart**的**EIGRP**配置中，使用数值为**5**的差异变量来实现非等价的负载均衡



在路由器**Earhart**上将**variance**指定为**5**以后，它的路由表就包含了第二条代价较高的路由（参见示例7-20）。在非等价负载均衡中的路由经常会碰到以下3种情况：

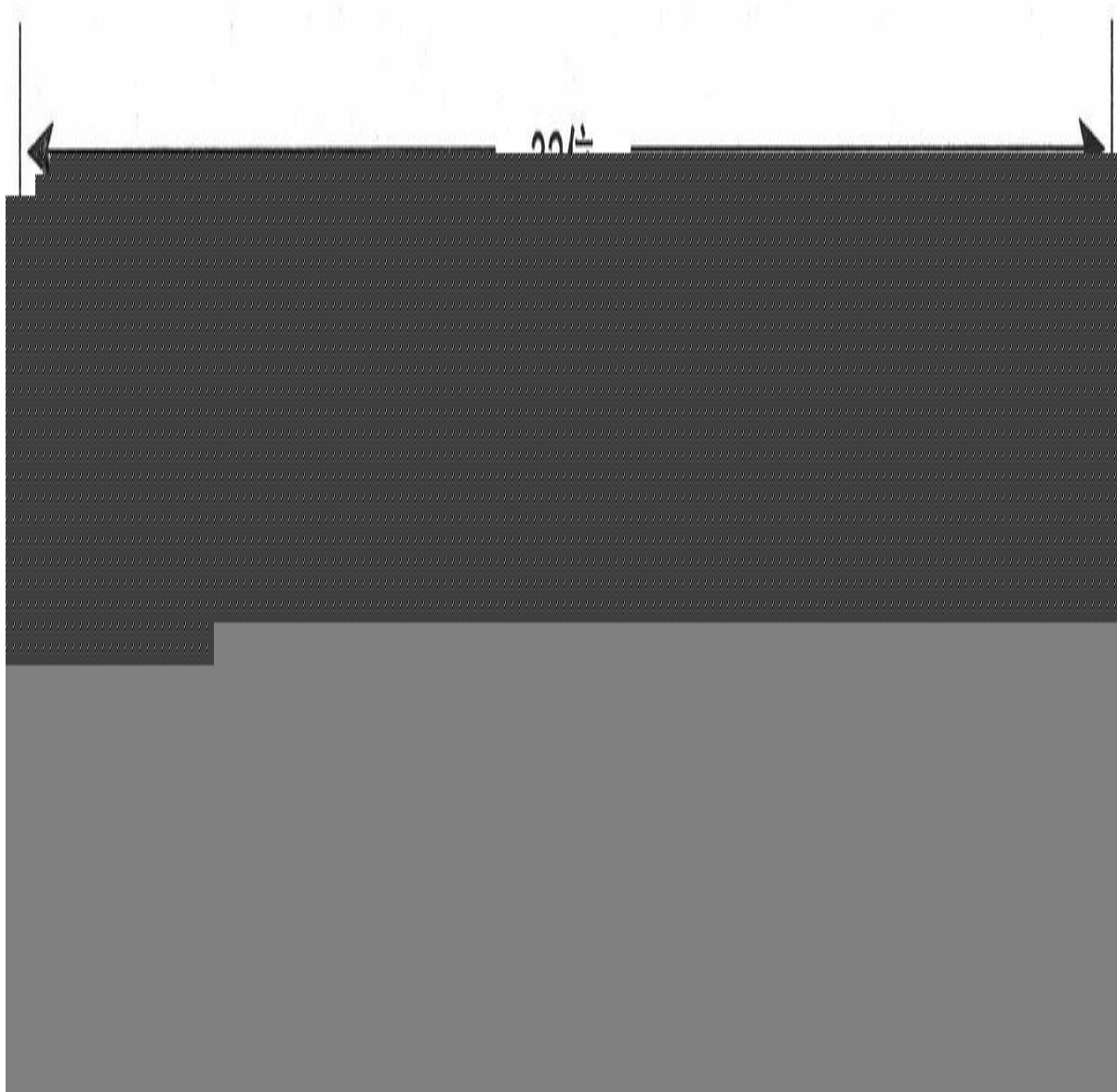
- 增加到负载共享“组”中的路由条数不能超过最大路径条数（**maximum-paths**）的限制。
- 下一跳路由器必须在度量值上更接近目的网络。换句话说，在下一跳路由器上到达目的网络的度量值必须小于本地路由器到达该目的网络的度量值。到达目的网络更近的下一跳路由器，通常被称为下游路由器（**downstream router**）。
- 最小路由代价的度量值乘以**variance**后，必须大于所增加的非最小路由代价的度量值。

示例7-20 从路由器**Earhart**到达网络**192.168.17.0**的第二条路径的复合度量值是**10514432**，或者说是最小代价的路由度量的**4.8**倍。如果**variance**的值设置为不小于**5**的话，那么**EIGRP**协议将把第二条路径加入到路由表中



关于按每目的地（per destination）和按每数据包（per packet）进行负载均衡的规则，已经在第3章中讨论过了，同样也适用于这里。如果数据包转发是快速交换（fast swithing）或缺省配置的CEF交换，就按照每目的地进行负载均衡；如果数据包转发是处理交换（process switching）或更改的CEF交换，就按照每数据包进行负载均衡。示例7-21显示了从路由器Earhart发出20个ping包的调试输出结果；在这里，CEF和快速交换已经通过命令**no ip cef** 和**no ip route-cache** 关闭了，因此路由器将执行按每数据包的非等价负载均衡。每5个数据包通过1544kbit/s的链路（下一跳是172.20.15.6）发送过后，就会有1个数据包通过256kbit/s的链路（下一跳是172.20.15.10）发送，与这两条路径大约5：1的度量比值是相一致的。

示例7-21 路由器执行的是按每数据包的负载均衡，即每通过低代价的链路发送**5**个数据包，就会通过高代价的链路发送**1**个数据包



如果**variance** 设置为1，那么EIGRP将只会把到达目的网络最小代价的路由加入到它的路由表中。然而，在一些情况下，例如，有时为了减小收敛的时间或帮助故障排错，即使没有发生负载均衡，也要将所有可用的路由都加入到路由表中。所有的数据包应该使用路径代价最小的路由，只有当主路径失效时，才被切换到下一个最好的路径。这里有一个隐含的缺省命令**traffic-share balanced**（也就是说，这个命令存在，但是不需要保留在配置文件里面）。在路由表中存在多条路径时，为了使路由器只使用最小代价的路径，可以把缺省的配置改成**traffic-share min**。如果有多条相等的最小代价的路径，并且配置了**traffic-share min**，那么EIGRP将执行等价负载均衡。

7.4.3 案例研究：设置最大的路径数

EIGRP协议可以进行负载均衡的路由路径的最大条数可以用**maximum-paths** 命令来设置。在12.3（2）T版本和后来发布的12.3（T）版本的IOS软件中，路径条数可以是1~16之间的任何值，而在这之前的早期版本中路径条数可以是1~6之间的任何值。所有版本的缺省值是4。

图7-33显示了从路由器Earhart到达网络172.18.0.0的3条不同代价的并行路由。网络管理员可能仅仅希望在这些路由中的两条路由上进行负载均衡，只有当这些路由中的任何一条被确认失效时，才使用第三条路由替代它。



图7-33 同时配置**maximum-paths**和**variance**命令，用来在路由器Earhart和Johnson之间3条链路中的其中两条上进行负载均衡。如果任何一条链路失效，第三条链路将替代失效的链路

从路由器Earhart上来看这3条链路的度量值分别是：

- 通过S0.4接口： $256 \times (9765 + (2000 + 10)) = 3014400$ ；
- 通过S0.5接口： $256 \times (19531 + (2000 + 10)) = 5514496$ ；
- 通过S0.6接口： $256 \times (78125 + (2000 + 10)) = 20514560$ 。

通过S0.6接口的度量值是最小代价路径度量值的6.8倍，因此，**variance**

的值应该是7。路由器Earhart的EIGRP的配置参见示例7-22所示。

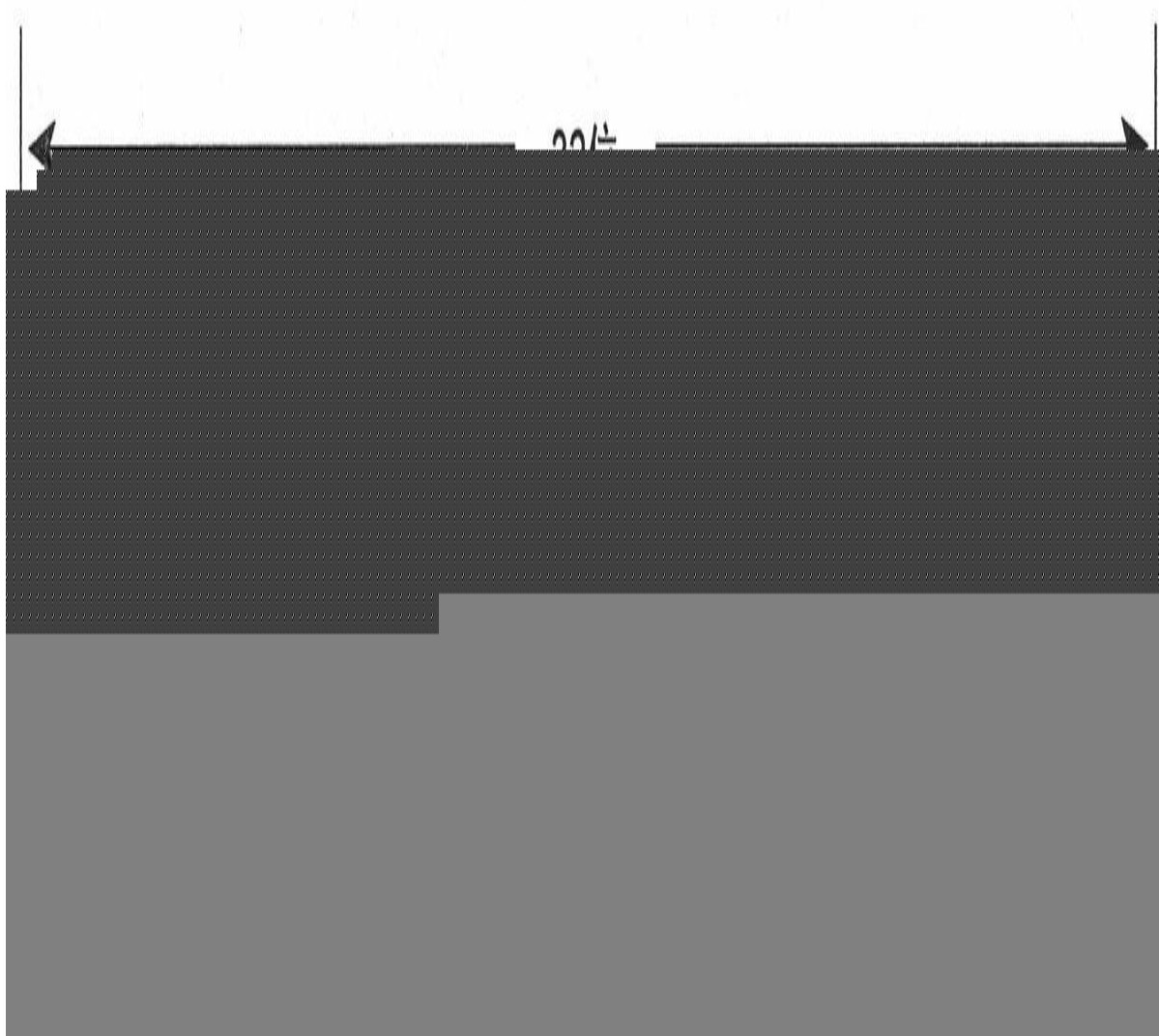
示例7-22 路由器Earhart的配置。使用**variance**命令和**maximum-paths**命令实现指定最大路径数的非等价的负载均衡



Variance 命令确保到达网络172.18.0.0的3条路由都是可用的；**maximum-paths** 命令用来限制负载均衡“组”中最多只有两条最佳的路由。在示例7-23中可以看到这种配置的结果。第一个路由表显示了路由器Earhart在3个度量中代价最低的两个链路上（即通过S0.4和S0.5接口的两条链路）进行负载均衡的情形。在通过S0.4的链路失效后，第二个路由表中显示了路由器在通过S0.5和S0.6的两条链路上进行负载均衡的情形。在每一种情况下，路由器都是执行和这两条路径的度量值成反比的负载均衡。

示例7-23 路由器Earhart的路由表显示了同时使用**variance**和**maximum-paths**命令配置到达网络172.18.0.0的负载均衡时，3条链路的某一条链路失效前后的不同结果





在网络中配置并行的路径时应该小心谨慎。太多的并行路径在某个链路失效时会增加EIGRP收敛的时间，这是因为邻居路由器的数目也增加了，因此增加了查询的范围。

7.4.4 案例研究：多个EIGRP进程

在上述案例的网络中增添两台新的路由器Bleriot和Post。在网络中创建两个EIGRP进程域，这两个进程域之间不进行通信。图7-34显示了这两个自主系统及其相关联的链路。

路由器Post、Johnson、Lindbergh和Earhart的配置都是相当简单的，路由器Johnson、Earhart和Lindbergh运行EIGRP 15，而路由器Post将运行EIGRP 10。在路由器Bleriot上的配置参见示例7-24所示。

示例7-24 路由器Bleriot上配置了EIGRP 15和EIGRP 10



每个EIGRP进程都只在指定网络的接口上运行。在路由器Cochran上，除了FA0/0之外的所有接口都属于网络172.20.0.0，参见示例7-25所示。

使用**passive-interface** 命令是为了防止EIGRP Hello发送到不属于它们的数据链路上去。请注意，由于路由器Cochran的接口属于网络172.20.0.0，因此**passive-interface** 命令可以用来限制不必要的路由选择协议流量。对于EIGRP协议来说，这个命令阻塞了不必要的Hello消息。这样将不会形成邻接关系，也不会发送其他的EIGRP流量。

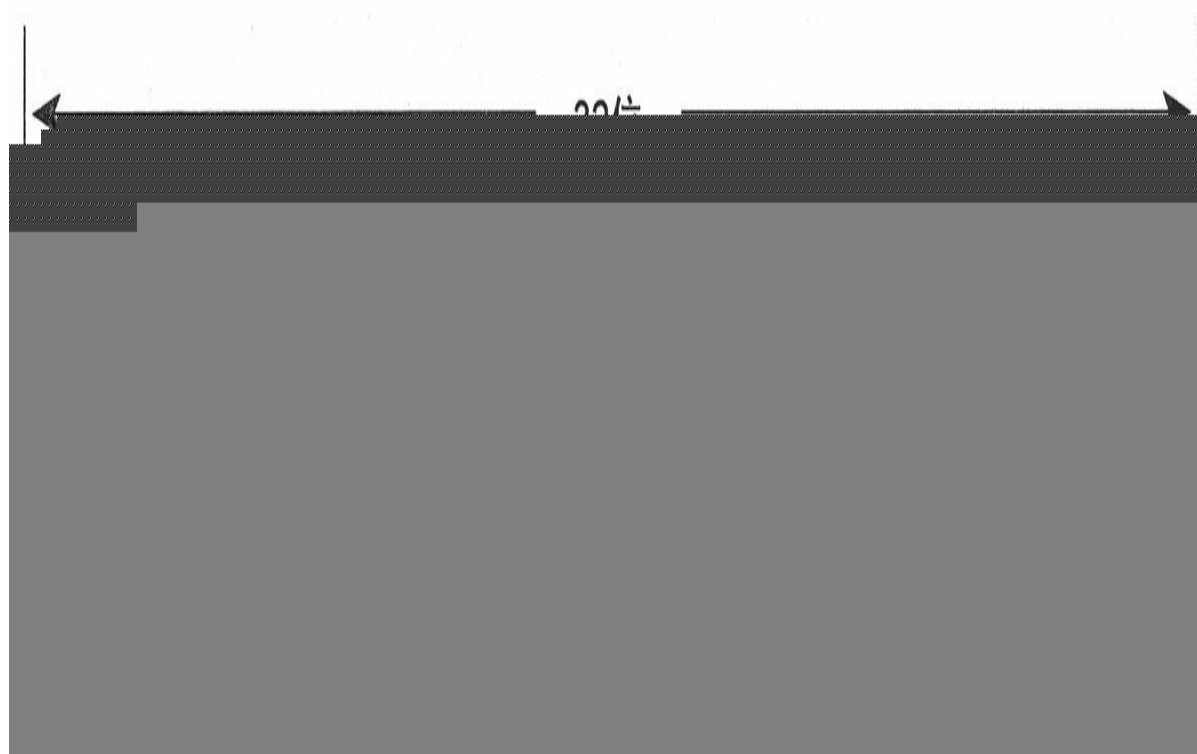
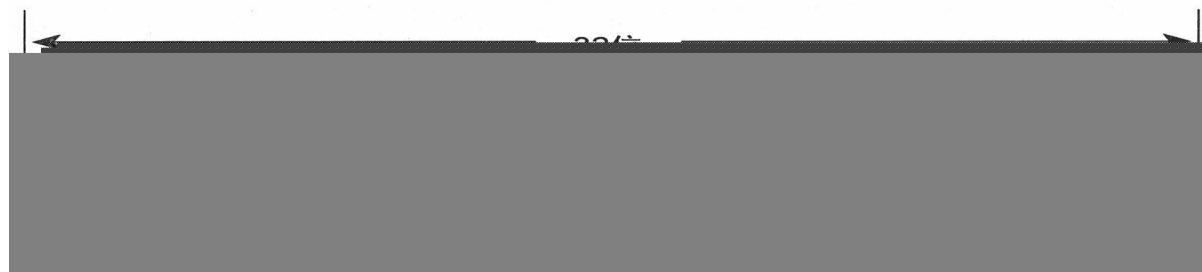


图7-34 路由器Bleriot和Cochran上分别运行多个EIGRP进程，以便使用这个IGP协议创建各自的自主系统（AS 10和AS 15）

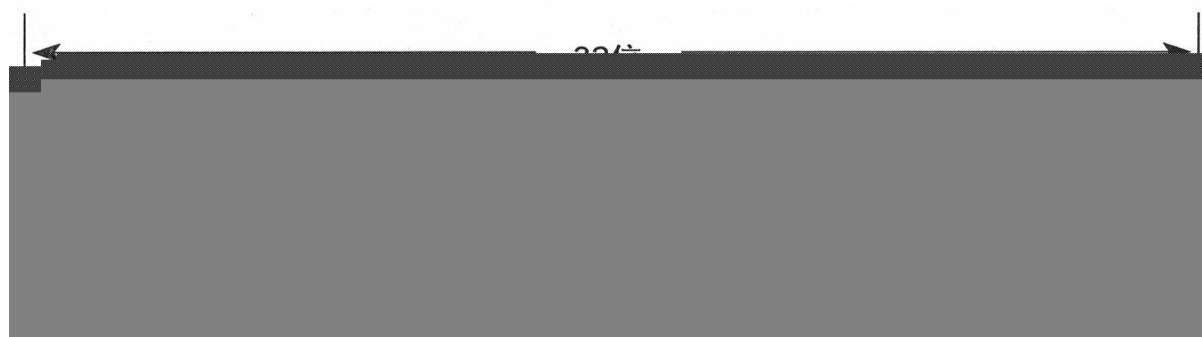
示例7-25 在路由器Cochran的EIGRP 15和EIGRP 10配置中要求配置

被动接口，这是由于**Cochran**相连的两个接口上使用了相同的主网络号



参见示例7-26所示，路由器Bleriot的邻居表显示了EIGRP 10进程的一个邻居10.108.16.2和EIGRP 15进程的一个邻居172.20.33.1。

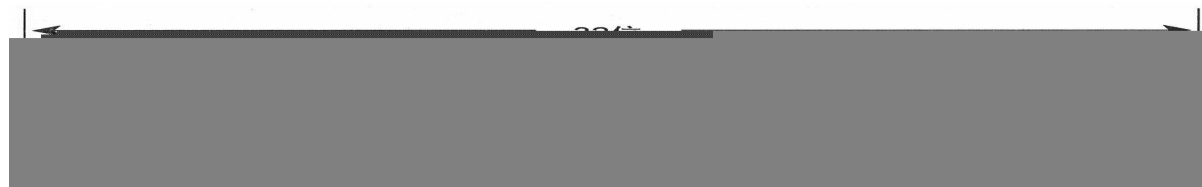
示例7-26 使用命令**show ip eigrp neighbor**可以显示与每个**EIGRP**进程相关联的邻居



在使用被动接口的地方，EIGRP的网络语句可以配置通配符掩码位。这里的通配符掩码位指定了在EIGRP进程中标识所包括的接口时使用的该地址的位数。

路由器Cochran的配置变化参见示例7-27所示。

示例7-27 在路由器**Cochran**的**EIGRP 15**和**EIGRP 10**的配置中，使用了该网络的通配符掩码位来缩小运行给定**EIGRP**进程的接口范围



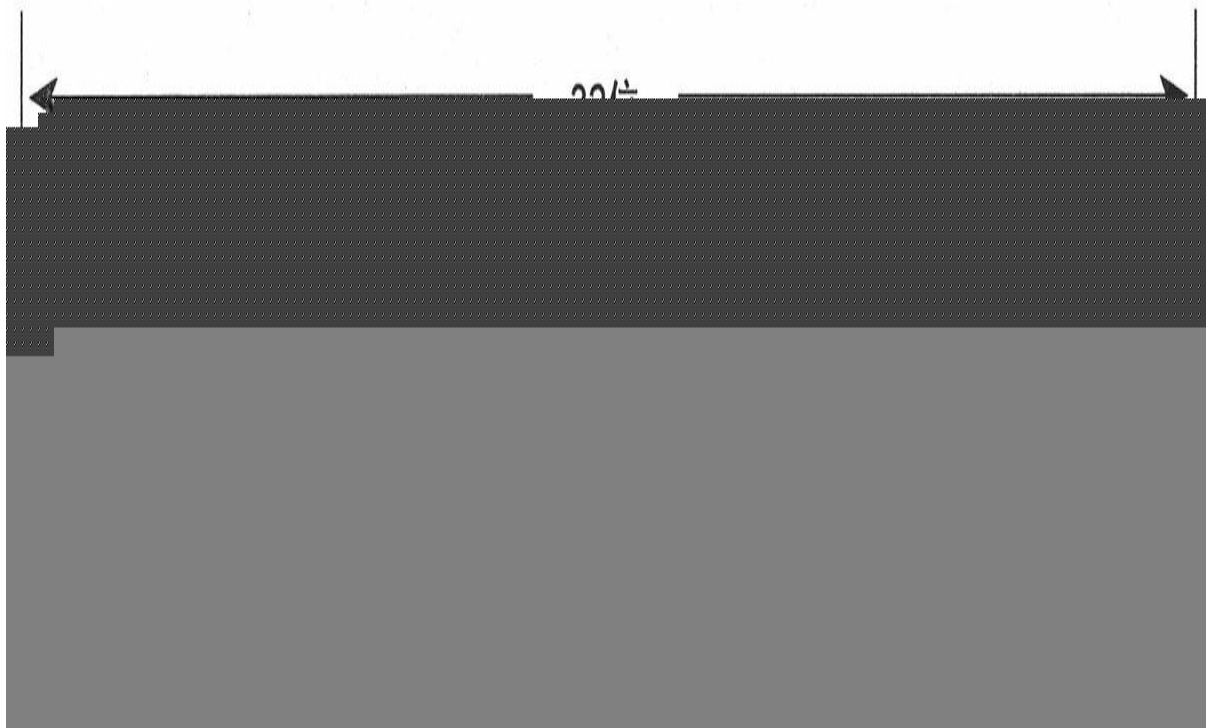
参见示例7-27，在路由器Cochran的配置中，指定了前3个八位组为172.20.15的地址的接口运行EIGRP 15，而前3个八位组为172.20.32的接口运行EIGRP 10。

7.4.5 案例研究：关闭自动路由汇总

在缺省的情况下，EIGRP协议和前面章节所讲述的协议一样，在网络边界上进行路由汇总。但是和前面那些协议不同的是，EIGRP的自动路由汇总可以被关闭。

示例7-28中显示了路由器Bleriot的路由表。这里请注意，地址172.20.32.0的路由是通过接口FE0/0学习到的，但这条路径却是经过了路由器Bleriot的快速以太网链路和一条从路由器Earhart到Cochran的串行接口。而经过路由器Post，与接口FE0/1相连的路径到达该目的地仅仅只有一个快速以太网链路的度量值。更仔细地查看这个路由表，我们发现经过FE0/1接口学习到的路由地址除了10.108.16.0，只有172.20.0.0/16这一条路由地址了。在经过地址10.0.0.0处的网络边界后，通告给路由器Bleriot之前，地址172.20.32.0已经被汇总。为了使路由器Bleriot能够通过路由器Post转发数据包到172.20.32.0，可以使用命令**no auto-summary**在路由器Post上关闭自动路由汇总。

示例7-28 在某些情况下，EIGRP的自动汇总会引起次优化的路由选择



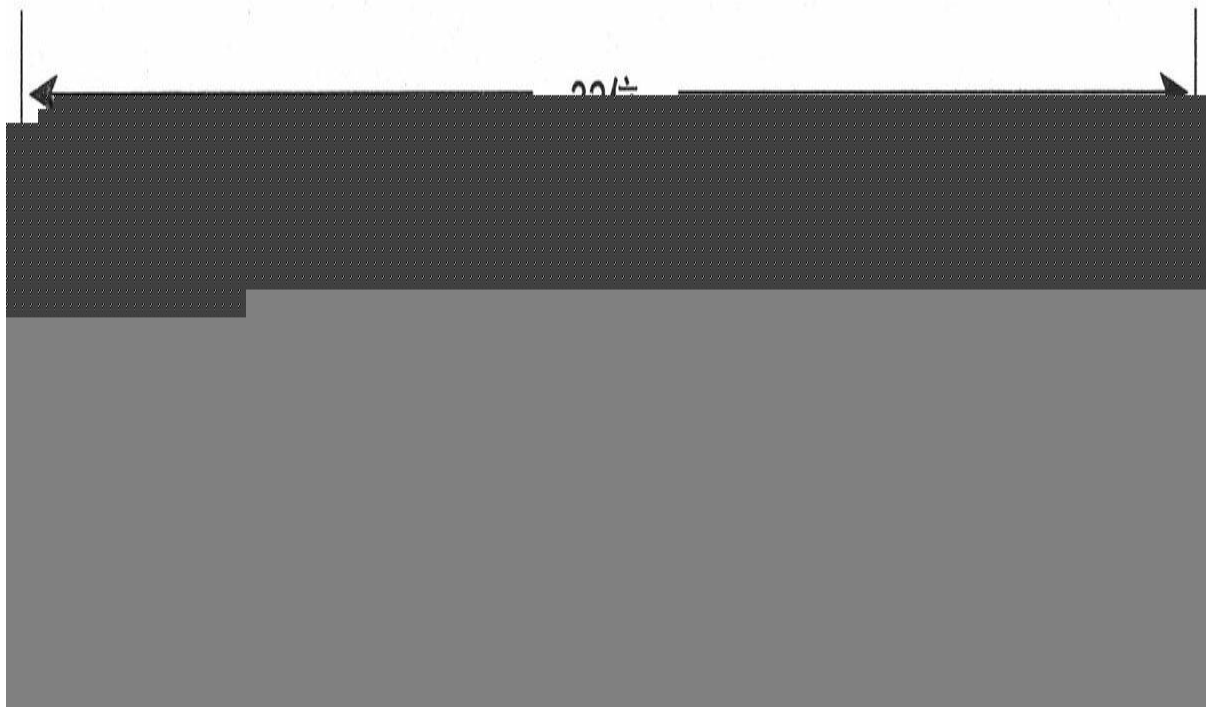
路由器Post的配置参见示例7-29所示。

示例7-29 路由器**Post**的配置中关闭了**EIGRP**的自动路由汇总功能。



路由器Bleriot新的路由表参见示例7-30所示。

示例7-30 在关闭了**EIGRP**的自动路由汇总后，在路由表中就可以看到远端地址的子网了



在图7-35中显示了另外一种情况，说明关闭路由汇总是有用的。



图7-35 在路由器Cochran和Lindbergh上关闭自动路由汇总功能来防止到达网络192.168.18.0的不确定的路由

在路由器Cochran和路由器Lindbergh上分别添加了新的以太网链路，并

且给它们分配的地址形成了不连续的子网。这两台路由器的缺省行为是把它们自己当作主网络192.168.18.0和172.20.0.0之间的边界路由器。结果，路由器Earhart将在它的两个串行接口上收到到达网络192.168.18.0的汇总路由通告。这个结果会产生一个路由选择不明确的情况：路由器Earhart记录了到达网络192.168.18.0的两条等价路径，要到达那些以太网之一的数据包可能会、也可能不会被路由到正确的链路上去。

使用命令**no auto-summary** 来关闭自动汇总功能后，路由器Lindbergh的配置参见示例7-31所示。

示例7-31 路由器Lindbergh的配置关闭了自动汇总功能



在路由器Lindbergh和Cochran上关闭了自动路由汇总功能后，分离的子网192.168.18.24/29和192.168.18.128/25将可以被通告到网络172.20.0.0中去，这样就在路由器Earhart上排除了不确定的路由。

7.4.6 案例研究：末梢路由选择

现在回忆一下本章早些时候讲述的DUAL算法。当一台路由器的EIGRP拓扑表变坏的时候（可能是度量值变大了，可能是后继路由器不再可达了），如果没有可行后继可以到达该地址，那么这个路由条目将进入active状态，同时这台路由器会向它的所有邻居路由器发送查询数据包。如图7-36所示，假如路由器Earhart到路由器Yeager的链路中断了，路由器Earhart就会发送查询数据包到它的所有邻居，包括路由器Johnson和Lindbergh，以便找到具有到达路由器Yeager的路径的邻居。路由器Earhart收到它所发出的所有请求那个路由条目的查询响应之前，它不会在拓扑表中改变那个active状态的条目。如果路由器Earhart到达Lindbergh的链路又出问题了，而这时Earhart还没有收到有关查询路由器Yeager的地址的任何答复，那么即使路由器Earhart和Yeager之间的链路恢复正常，路由器Yeager的地址也会继续保持active状态。

如图7-36所示，在这个网络中，路由器Johnson和Lindbergh没有任何后门路由到达其它任何站点。它们在设计为星型（hub-and-spoke）拓扑的

网络中称为分支（spoke）路由器。这些路由器不用来为网络中的任何地址提供透传路径（transit path）。如果路由器Lindbergh或Johnson需要向非本地的站点地址转发数据包时，那么数据包就会被转发到路由器Earhart上。例如，路由器Lindbergh知道一条到达地址172.20.10.0的路径，这条路径是经过路由器Earhart的。为了减少网络不稳定的风险，不需要向路由器Johnson发送有关网络上其他目的地地址的查询。路由器Johnson和Lindbergh可以配置为末梢路由选择（stub routing）。

一台具有EIGRP末梢邻居的路由器将不会向它的末梢发送查询，因此这就排除了配置末梢的远端站点引起“卡在活动状态”的情形发生的机会，也降低了网络中其余部分路由选择的不稳定。

下面路由器Johnson被配置为一台EIGRP末梢路由器。Johnson有关末梢路由器的配置 参见示例7-32所示。

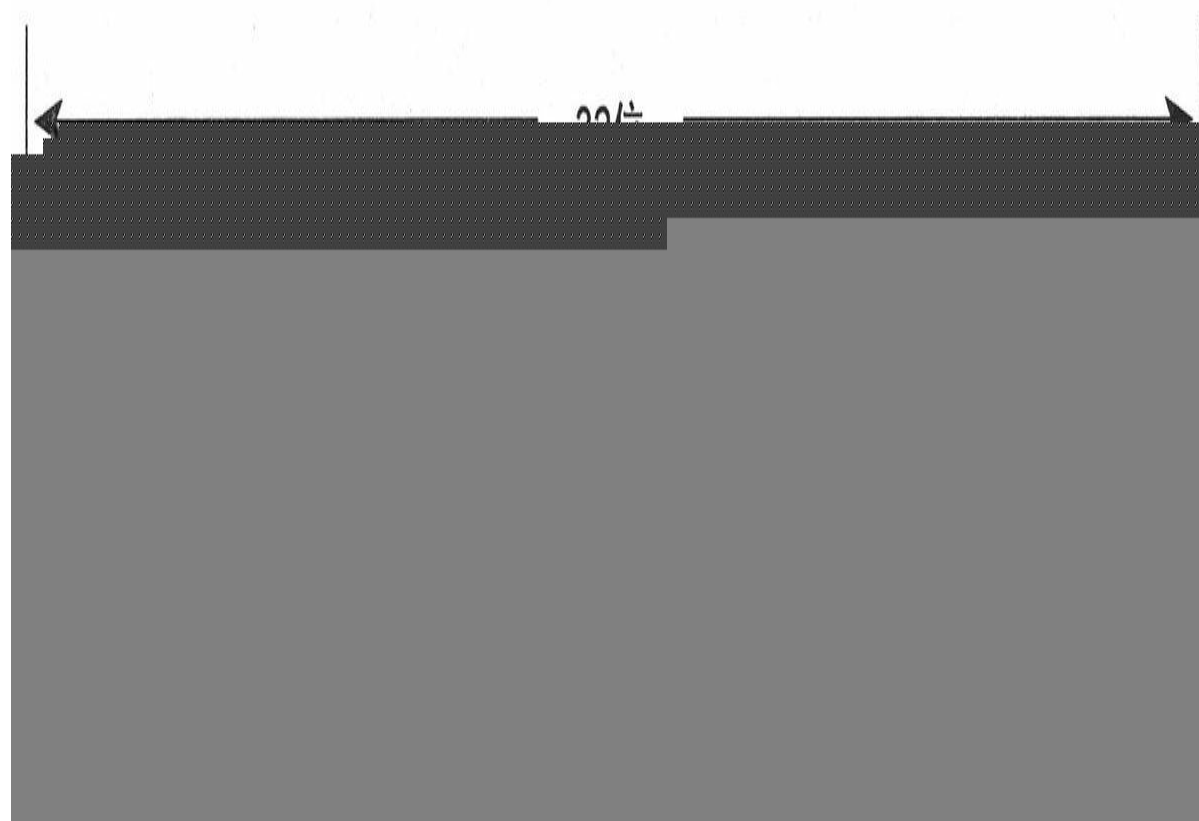


图7-36 路由器Yeager通过添加到网络中的单条链路连接到路由器Earhart

示例7-32 路由器Johnson的EIGRP末梢路由器配置



路由器Earhart作为中心路由器，不需要改变配置。

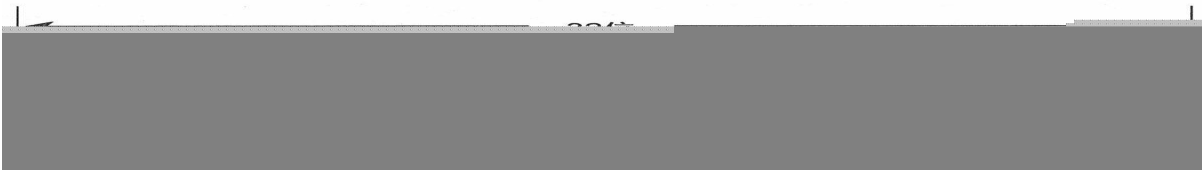
命令**eigrp stub** 会使路由器Johnson仅仅发送包含与它直连的路由和汇总路由的更新消息。路由器Johnson使用下面的命令可以配置包括下列路由的任何组合，如直连路由、汇总路由、静态路由，或者重新分配到EIGRP内的路由，等等：

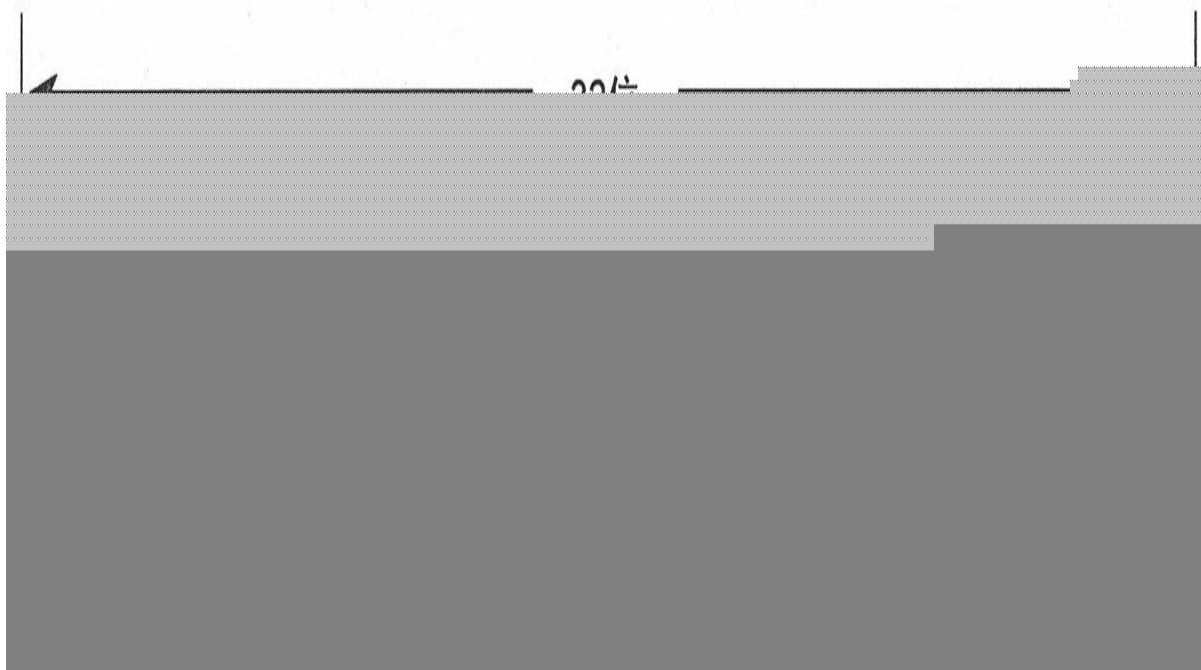
eigrp stub {connected | redistributed | static | summary | receive-only}

路由器Johnson也可以利用**receive-only** 选项配置为在更新消息中不发送任何路由信息。在使用**receive-only** 选项的配置下，远端的路由器在更新消息中将不包括任何地址。路由器Johnson将不得不利用其他方式把与它直连的地址通告到网络的其余部分，以便使流量可以到达那个地址的站点，这可以通过在路由器Earhart上配置静态路由来实现。

参见示例7-33所示，可以在中心路由器上使用命令**show ip eigrp neighbor detail** 检验配置为末梢路由器的邻居。路由器Earhart的输出信息显示路由器Johnson配置成为一台末梢路由器。

示例7-33 命令**show ip eigrp neighbor detail**显示了配置为EIGRP末梢路由器的邻居路由器





路由器Earhart具有3个末梢路由器。其中有3条链路连接到路由器Johnson: 172.20.15.22、172.20.15.14,以及172.20.15.17。

现在，假定为了增加与路由器Lindbergh相连的LAN通过网络核心区域的流量的冗余性，我们在路由器Lindbergh和Cochran之间增加一条新的链路，如图7-37所示。在路由器Lindbergh配置作为一台末梢路由器之前，如果路由器Cochran与Earhart之间的链路出现故障，路由器Cochran将会发送查询到路由器Lindbergh，希望在它的拓扑表中找到到达目的地地址的另一条路径，在这里实例中的地址是172.20.10.0。路由器Lindbergh将会响应一个确定的应答，因为在Lindbergh的拓扑表中具有一条经过路由器Earhart学到的网络172.20.10.0的路由条目。因此，从路由器Cochran到达172.20.10.1的流量将经过路由器Lindbergh。

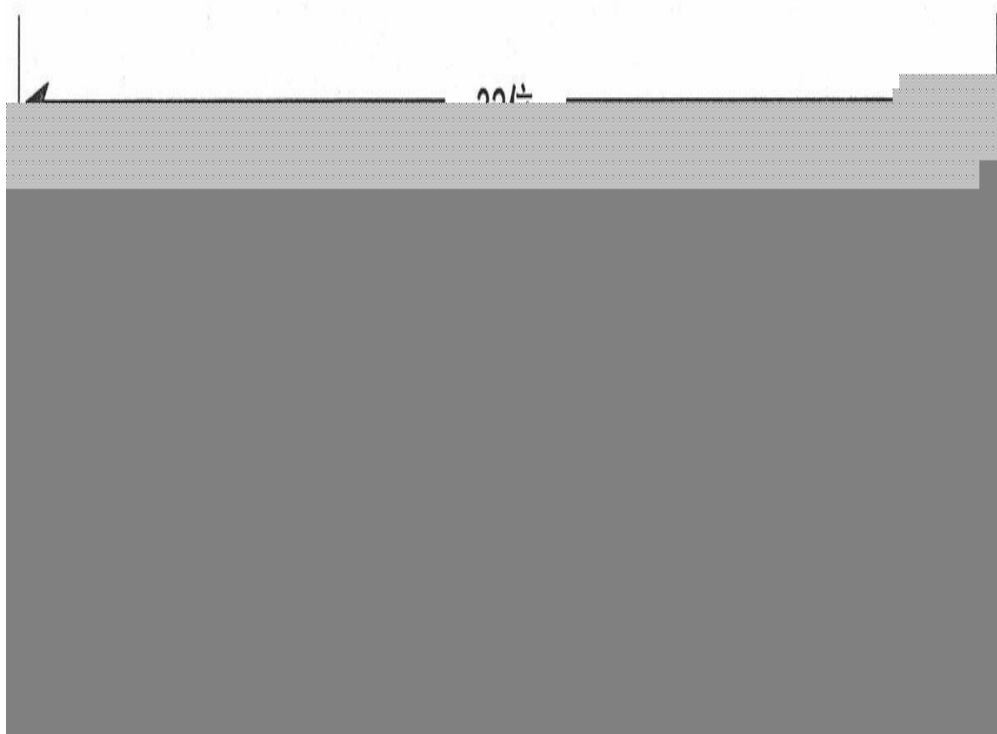
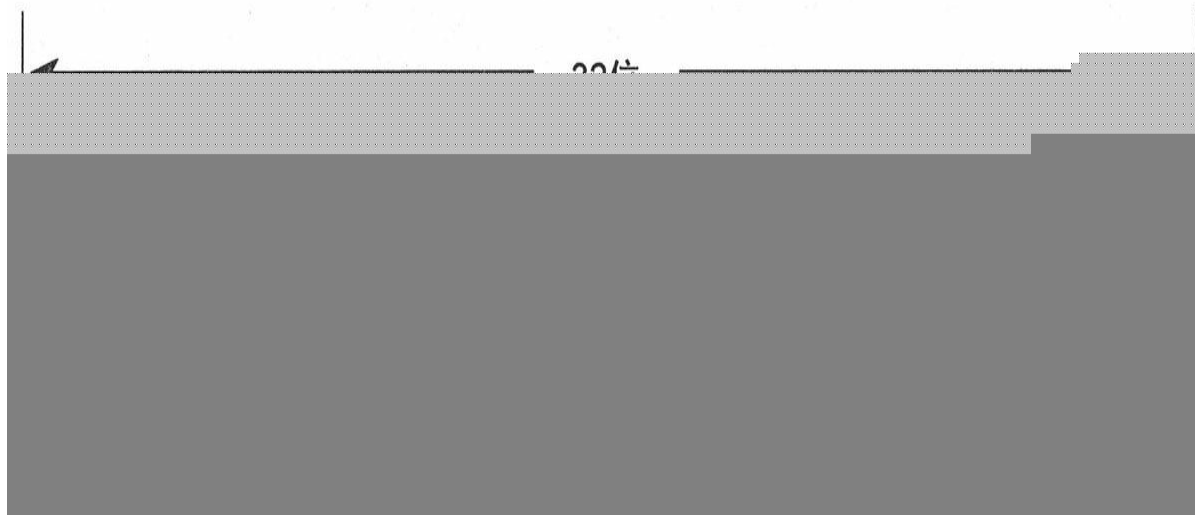


图7-37 在路由器**Lindbergh**与**Cochran**之间增加了一条新的链路，用来增加**Lindbergh**所连接的**LAN**网络到核心网络之间流量的冗余性

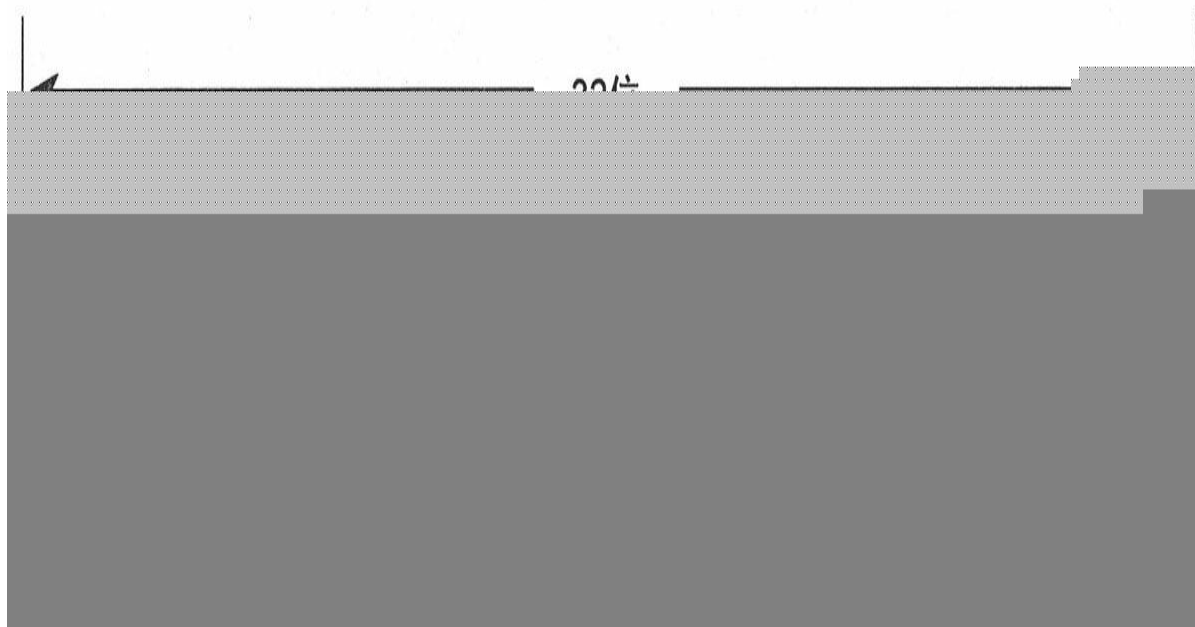
虽然这是另一条可选的路径，但是经过一个远程站点转发流量并不总是明智的选择。在这个实例中，路由器**Lindbergh**的那条链路是用来作为从它所连接的地址到核心网络地址之间流量的冗余链路的，而不是希望**Lindbergh**作为一个中间透传路由器。它的带宽也许不够用来提供中间透传的路由。末梢路由选择可以简单地解决这个问题。作为一台末梢路由器，没有任何查询消息发送到路由器**Lindbergh**上，因而路由器**Lindbergh**也不会使自己成为路由器**Cochran**的另一条可用的路径。而且，路由器**Lindbergh**只会发送包括以下这些路由的更新：直连路由、汇总路由、静态路由，或重新分配路由，而不会包括像172.20.10.0这样的远端路由。

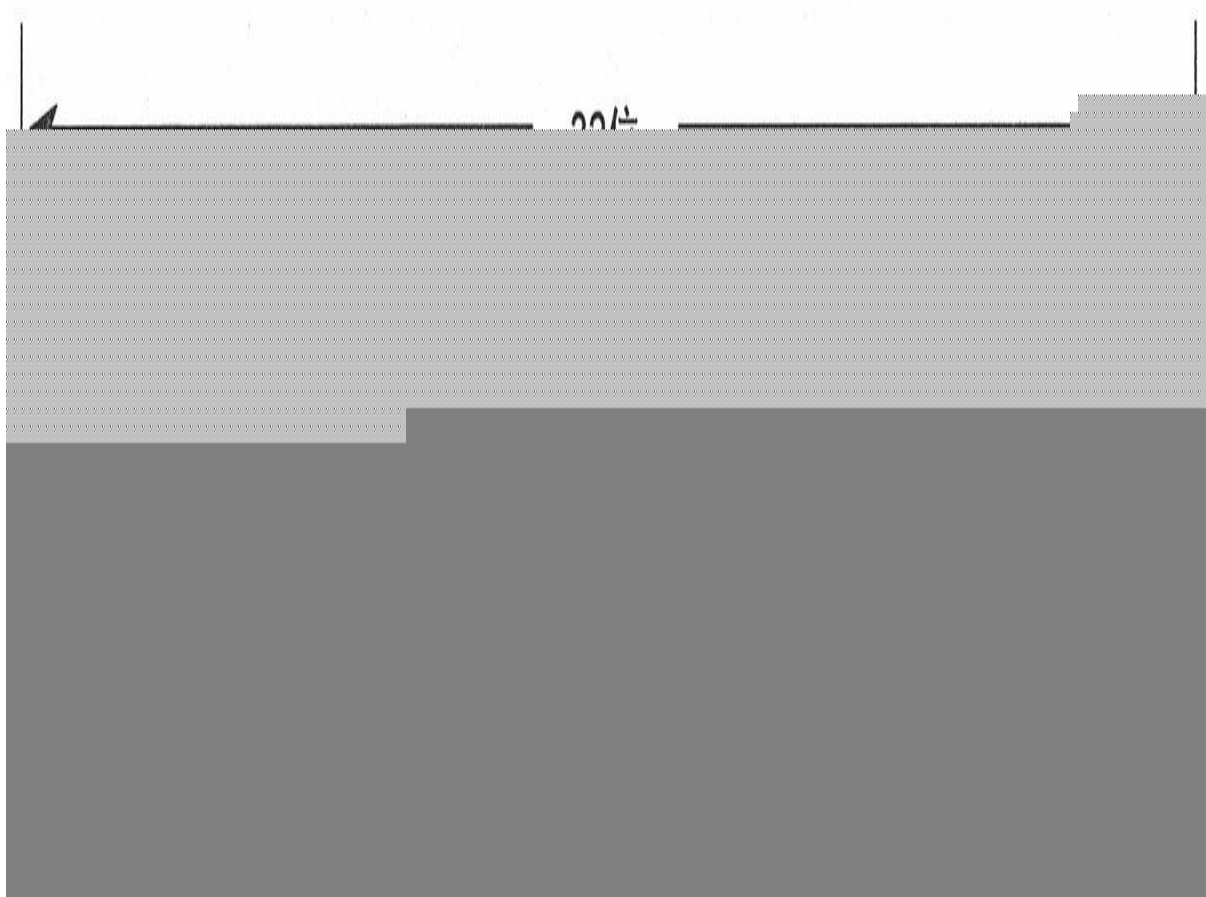
在示例7-34中显示了路由器**Lindbergh**（连接到**Cochran**的Serial0/0.4接口）没有配置为末梢路由器时，路由器**Cochran**的EIGRP邻居。假定路由器**Cochran**到路由器**Earhart**的两条链路都中断了。参见示例7-35，路由器**Cochran**随后的拓扑表显示所有地址都是经过路由器**Lindbergh**（Serial0/0.4）转发的。

示例7-34 路由器**Cochran**的**EIGRP**邻居表显示了它的邻居路由器。路由器**Lindbergh**不是一台末梢路由器



示例7-35 在路由器**Cochran**与**Earhart**之间的主要链路都发生故障后，所有的地址都通过具有双连接的分支路由器**Lindbergh**访问了





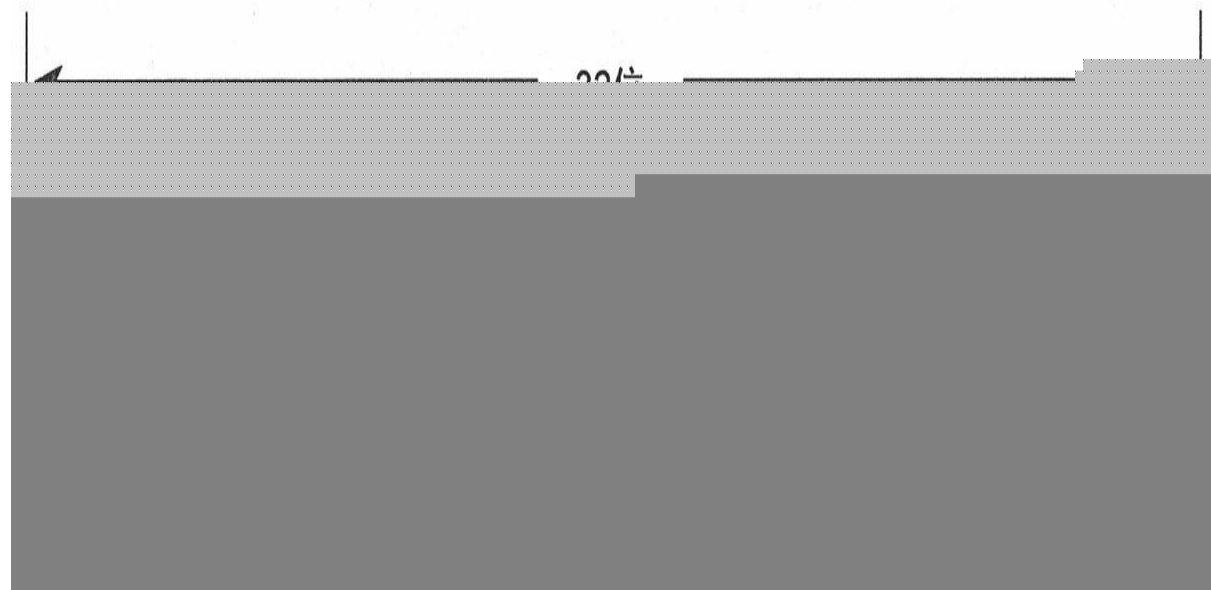
现在，将路由器Lindbergh配置为一台EIGRP的末梢路由器，参见示例7-36所示。

示例7-36 路由器Lindbergh配置为一台EIGRP末梢路由器

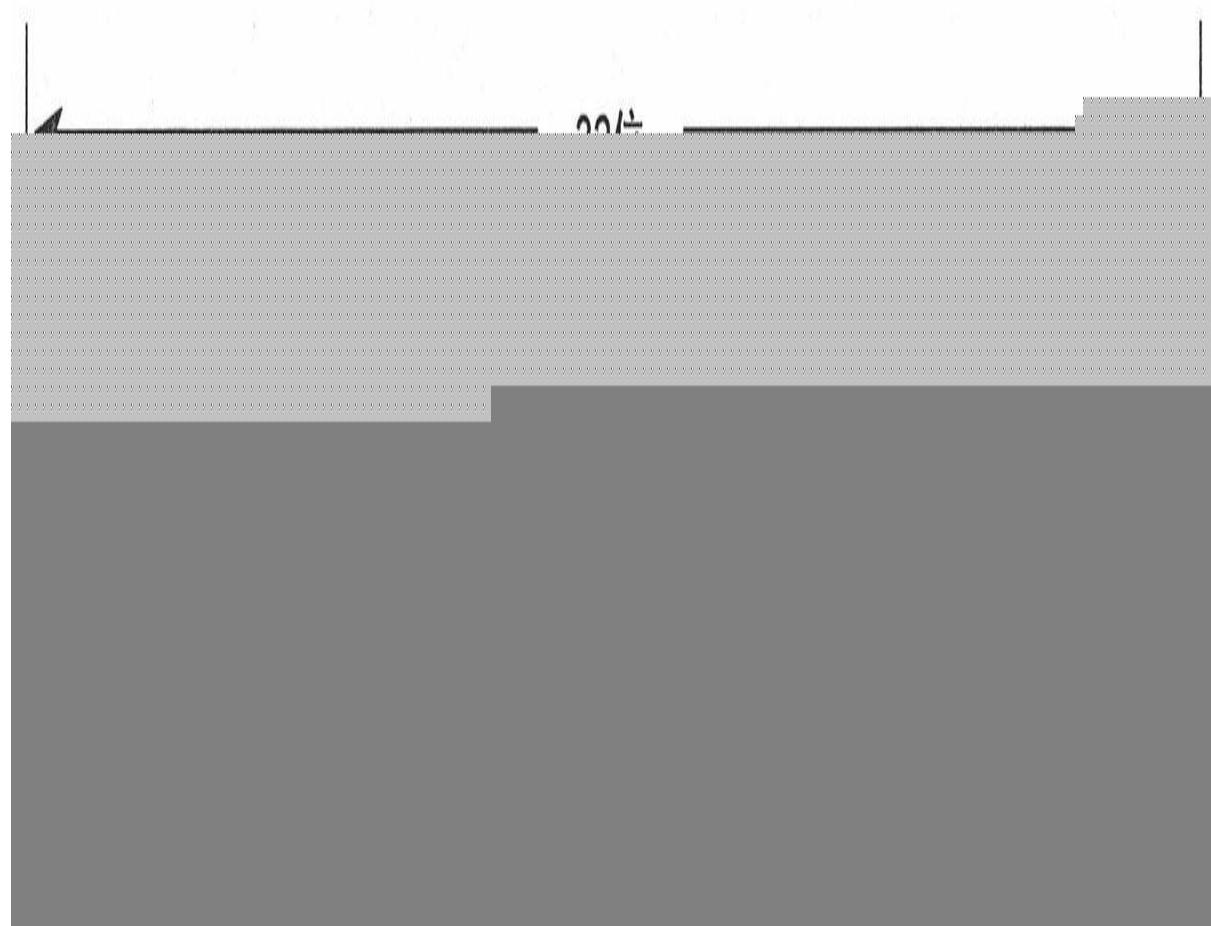


示例7-37显示了路由器Cochran的EIGRP邻居表，示例7-38显示了路由器Cochran到Earhart的两条链路再次中断后Cochran的EIGRP拓扑表。

示例7-37 路由器Cochran的EIGRP邻居表显示了它的邻居路由器。路由器Lindbergh是一台末梢路由器。路由器Cochran运行的是比Earhart更新的IOS版本（12.3）。请注意在这里被抑制的查询是明确表明的



示例7-38 在路由器**Cochran**与**Earhart**之间的主要链路都发生故障后，只有与路由器**Lindbergh**相连的地址可以通过串口**Seria10/0.4**到达



配置路由器Lindbergh作为EIGRP末梢路由器可以防止它在核心链路发生故障期间变成中间透传的路由器。

配置EIGRP的末梢路由选择可以最大限度地减少查询的数量，这样极大地增加了EIGRP网络的扩展性，而网络中断时间要求相关地址处于active状态。

末梢路由选择取消了发送到末梢路由器的查询，但是它并没有为末梢点隐藏网络其余部分的拓扑。路由器Earhart能够对末梢点隐藏网络其余部分的拓扑。由于到每个子网的所有数据包总是转发到中心路由器的，因此这些末梢路由器就不需要了解关于每个子网的信息。路由器Earhart可以通过汇总地址完成这项任务。

7.4.7 案例研究：地址汇总

在图7-37所示的网络中，路由器Yeager具有单条链路连接到路由器Earhart上。路由器Earhart必须通告到路由器Yeager的6个地址能够被汇总为两条聚合的地址。路由器Earhart的配置参见示例7-39所示。

示例7-39 路由器Earhart的配置汇总路由到路由器Yeager



命令**ip summary-address eigrp**将会自动地抑制更具体的到达路由器Yeager的网络的通告。示例7-40显示了配置聚合地址前后路由器Yeager的路由表。即使在这么一个小的网络中，通过EIGRP学到的路由条目数也减少了一半；在一个大型网络中，压缩路由表的大小和存储它们所需要的内存是很重要的。

示例7-40 在路由器Earhart上配置汇总地址前后所显示的路由器Yeager的路由表

7.4.8 认证

MD5加密校验和是EIGRP协议唯一支持的认证方式，初看起来，这和RIPv2或OSPF协议相比灵活性较差，因为后两种协议同时支持MD5认证和明文口令认证。然而，明文口令认证应该只使用在邻居设备不支持比较安全的MD5认证的时候。由于EIGRP协议仅会在两台Cisco设备之间互相宣告，因此这种情况不会发生。

配置EIGRP协议的认证有以下几个步骤：

步骤1： 定义一个带有名字的钥匙链。

步骤2： 在钥匙链上定义一个或一组钥匙。

步骤3： 在接口上启用认证并指定使用的钥匙链。

步骤4： 可选地配置钥匙的管理。

钥匙链的配置和管理已经在第6章中讲述过。EIGRP协议认证是在接口上配置命令**ip authentication key-chain eigrp** 和命令**ip authentication mode eigrp md5** 来启用和连接到一个钥匙链上的。[\[19\]](#)

参考图7-37，在路由器Cochran到路由器Earhart的接口上启动EIGRP认证的配置参见示例7-41所示。

示例7-41 配置路由器Cochran与路由器Earhart使用MD5认证



在路由器Earhart上也要作相似的配置。关于钥匙链的管理命令**accept-lifetime** 和**send-lifetime** 的使用方法已经在第6章中讲述过了。

7.5 EIGRP故障诊断

RIP协议的路由信息交换的故障诊断是一个相当简单的过程。路由选择更新要么传播出去了要么没有传播出去，要么包含了精确的路由信息要么没有包含。EIGRP协议复杂性的增加也意味着故障排除过程的复杂性增加了。在EIGRP协议的故障排除中，必须验证邻居表和邻接关系的正确性，检查DUAL算法的查询/响应过程的正确性，并考虑在自动汇总时对VLSM的影响。

本节的案例研究讲述了一系列典型的事件，可以用来追踪一个EIGRP的故障。随后的一个案例研究将讲述在一个大型的EIGRP网络中引起网络不稳定的一些偶然原因。

7.5.1 案例研究：邻居丢失

图7-38中显示了一个小型的EIGRP网络。用户抱怨子网192.168.16.224/28无法到达。

仔细察看路由表，发现在路由器Grissom上有一些错误（参见示例7-42所示）。[\[20\]](#)

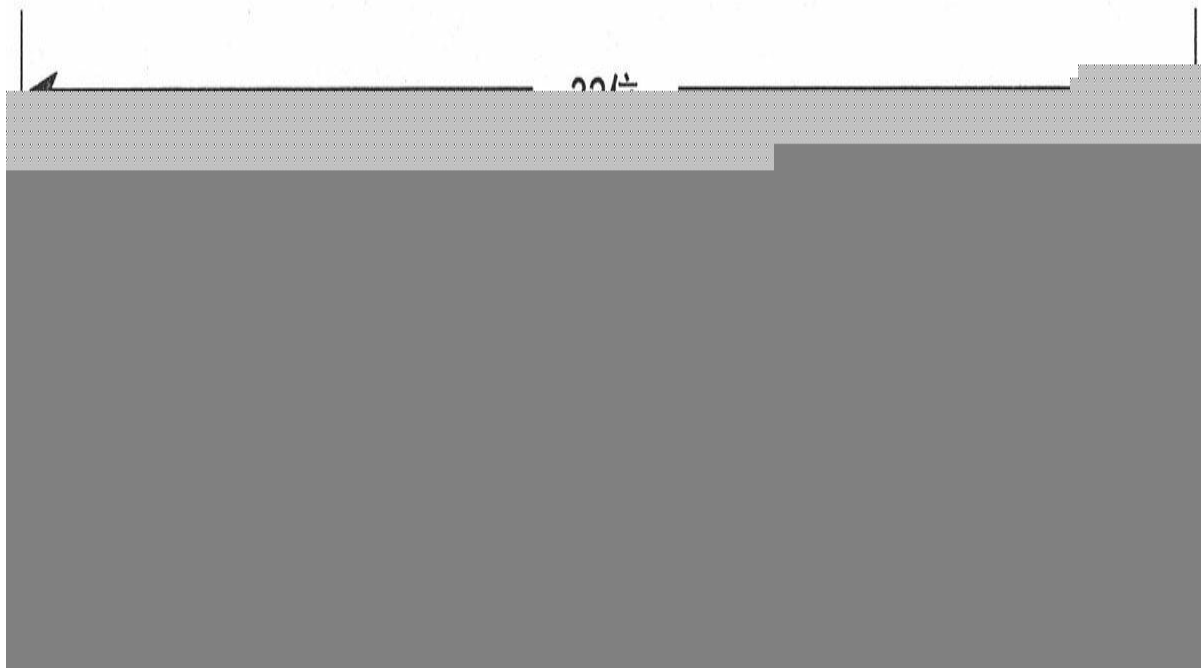
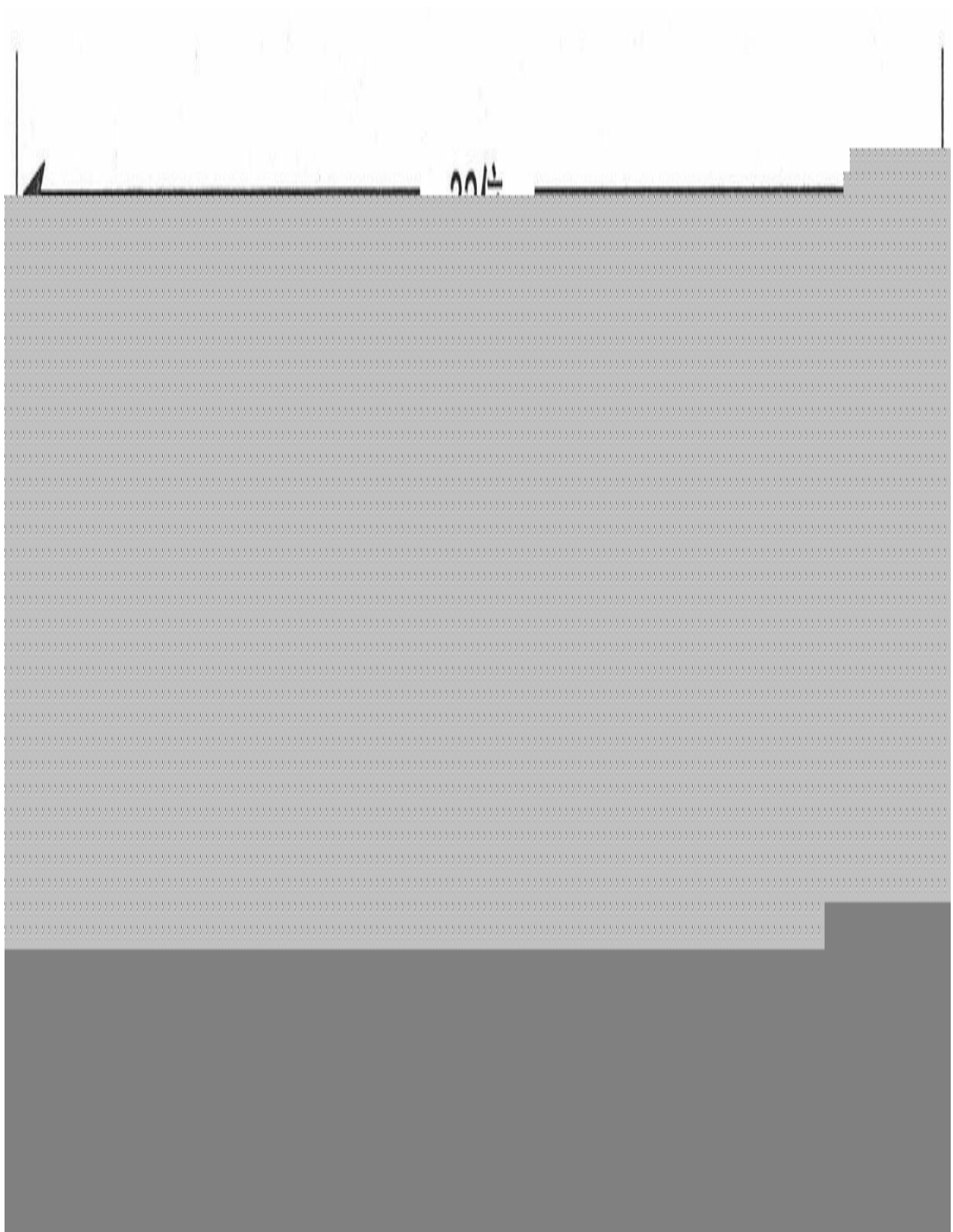


图7-38 在这个例子的**EIGRP**网络中，通过路由器**Grissom**不能到达子网**192.168.16.224/28**

示例7-42 路由器**Shepard**和**Grissom**的路由表显示了路由器**Grissom**的**EIGRP**进程没有通告和接收到关于子网**192.168.16.16/28**的路由



从示例7-42中的两个路由表中可以得到以下的信息：

- 路由器Shepard的路由表中没有子网192.168.16.40/30和子网192.168.16.224/28的信息，虽然路由器Grissom的路由表中含有这些信息。
- 路由器Grissom的路由表中没有包含任何应该由路由器Glenn或Shepard通告的子网。
- 路由器Shepard的路由表中包含了由路由器Glenn通告的子网（并且路由器Glenn的路由表中包含了路由器Shepard通告的子网，虽然它的路由表没有包括在示例7-42中）。

从以上的观察信息可以得出结论，路由器Grissom没有正确地接收和通告关于子网 192.168.16.16/28的路由。

在这些可能引起故障的原因中，应该首先来察看一些最简单和直接的原因。这些原因有：

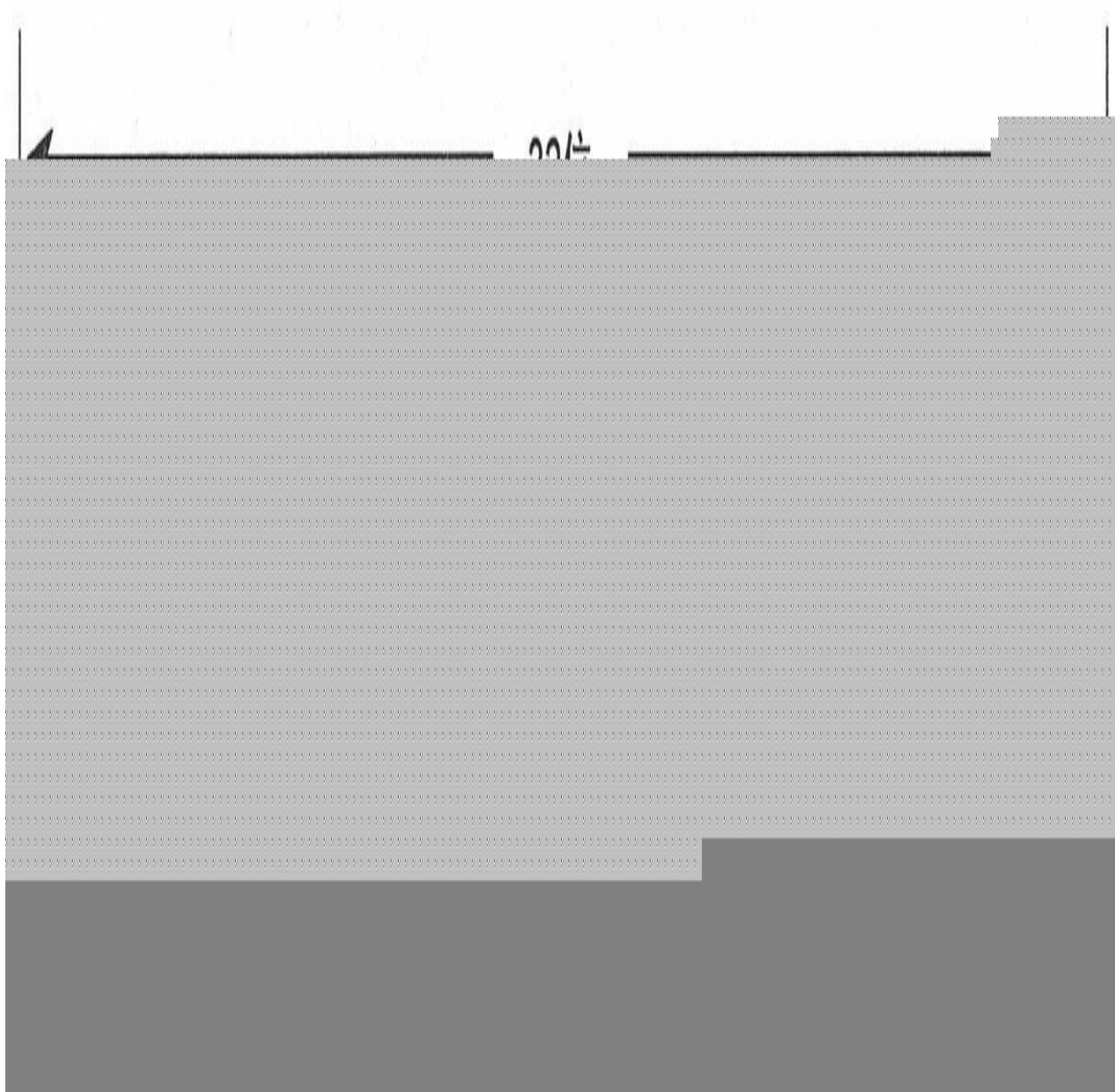
- 接口地址或者掩码配置不正确；
- EIGRP的进程ID号不正确；
- network语句遗漏或者不正确。

在这个实例中，没有发现EIGRP或地址的配置错误。

接下来，仔细检查一下邻居表，分别在路由器Grissom、Shepard和Glenn上察看邻居表，参见示例7-43所示，有两个事实比较明显：

- 路由器Grissom（192.168.16.19）在它的邻居的邻居表中，但是它的邻居却不在路由器Grissom的邻居表中。
- 整个网络已经运行了5个多小时了，这个信息可以从除了路由器Grissom外的所有邻居的uptime统计信息反映出来。然而，路由器Grissom的uptime显示大约是1min。

示例743 路由器Shepard和Glenn发现路由器Grissom是它们的邻居，但Grissom却看不到它们。这意味着路由器Shepard和Glenn可以接收来自Grissom的Hello消息，但是Grissom收不到路由器Shepard和Glenn。



假设路由器Grissom在路由器Shepard的邻居表中，则路由器Shepard必须接收来自路由器Grissom的Hello消息。然而，路由器Grissom显然不接收来自路由器Shepard的Hello消息。没有Hello消息的双向交换，就不能形成一个邻接关系，路由信息也就无法进行交换。

更仔细地检查路由器Shepard和路由器Glenn的邻居表，更加证实了这个假设：

- 路由器Grissom的SRTT为0，表示从来没有一个数据包在路由路

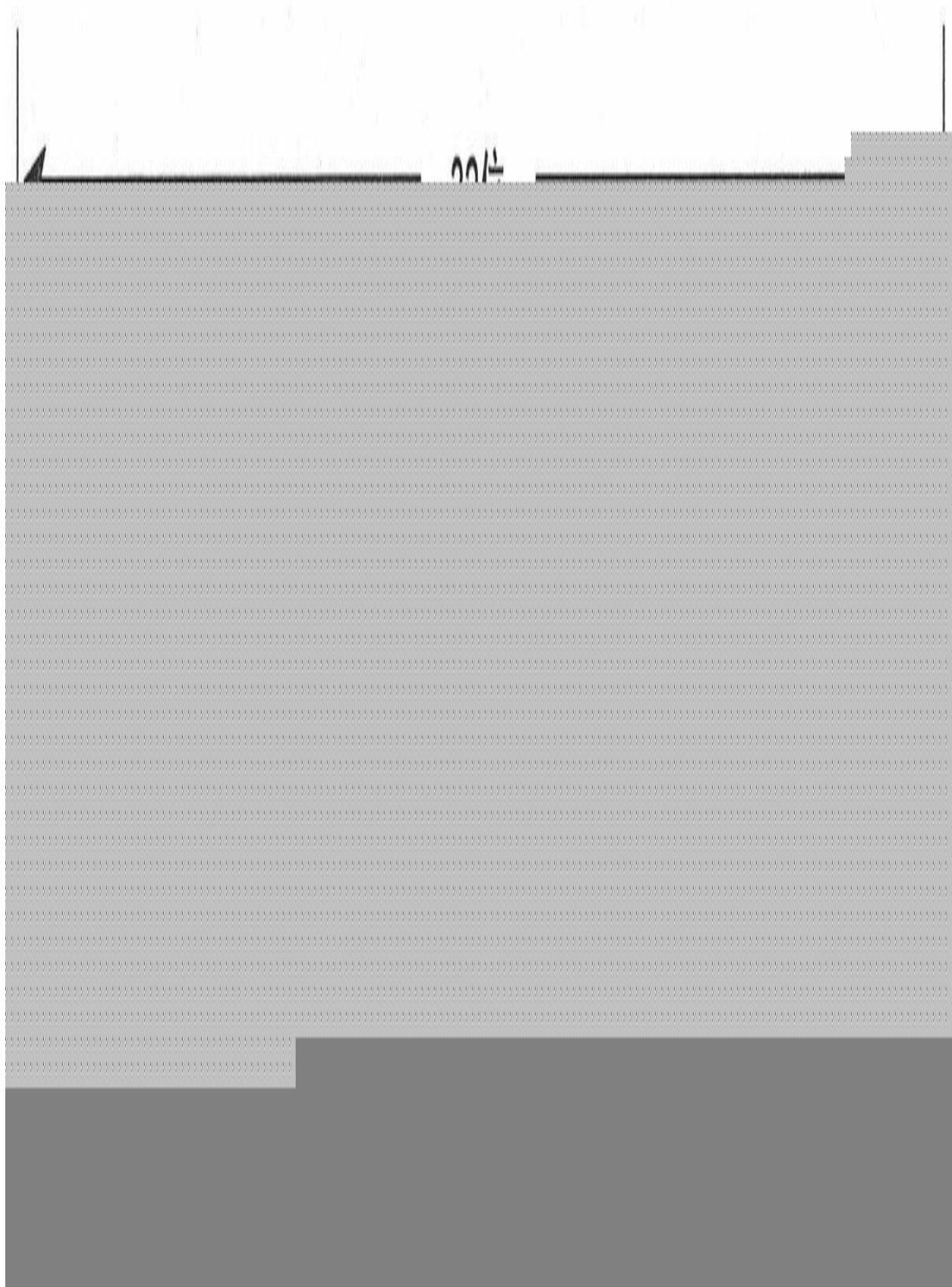
径上进行过往返（round-trip）。

- 路由器Grissom的RTO分别增加到了5s和8s。
- 有一个关于路由器Grissom的数据包在队列中等待发送。
- 关于路由器Grissom记录的序列号为0，表示从来没有从路由器Grissom接收到可靠的数据包。

这些因素都表明，这两台路由器都在试图向路由器Grissom发送可靠的数据包，但是却 没有收到一个ACK确认消息。

在示例7-44中，在路由器Shepard上使用命令**debug eigrp packets** 可以更好地观察到底发生了什么。这样的话，所有的EIGRP数据包类型都将显示出来了，但是可以使用第二个调试命令：**debug ip eigrp neighbor 75 192.168.16.19**。该命令是在第一个命令的基础上增加了一个过滤器。它告诉**debug eigrp packet** 只需要显示EIGRP 75（图7-38中那些路由器的进程ID号）的IP数据包并且只显示和邻居192.168.16.19（路由器Grissom）有关的那些数据包。

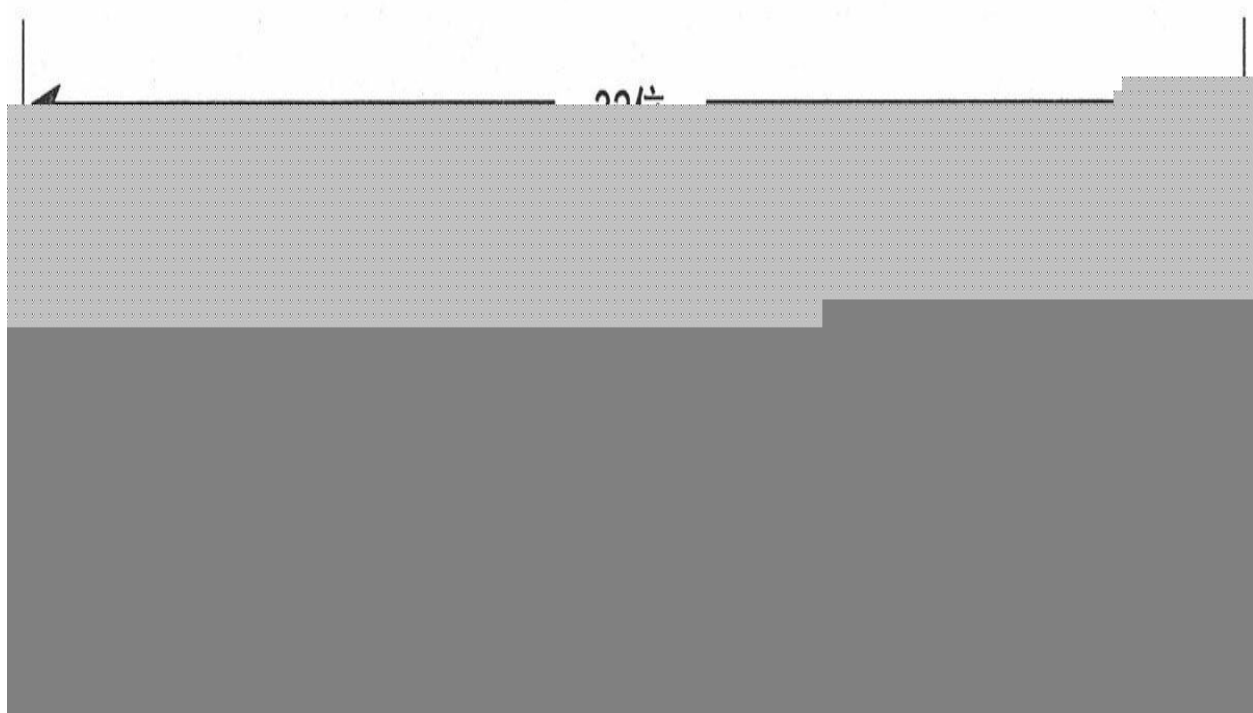
示例7-44 调试命令**debug ip eigrp neighbor**用来控制命令**debug eigrp packet**所显示的数据包



示例7-44中显示了路由器Shepard正在接收来自路由器Grissom的Hello数据包。它也显示了路由器Shepard正在试图发送更新消息给路由器Grissom,而路由器Grissom却不确认它们。这样进行了第16次重试后,将显示一条“重传尝试次数限制超出”(Retransmission retry limit exceeded)的消息。这个超出限制的计数可以从路由器的邻居表中看出,因为所显示的路由器Grissom的uptime时间都较低——一旦超过重传尝试次数的限制,路由器Grissom将从路由器Shepard的邻居表中删除。但是,由于路由器Shepard依然可以接收到来自路由器Grissom的Hello数据包,所以路由器Grissom又将在路由器Shepard的邻居表中迅速地再次出现,这个处理过程又将再次开始。

示例7-45显示了在路由器Shepard上调试命令**debug eigrp neighbors** 的输出信息。这个命令没有指定具体的IP地址,而是改为显示EIGRP的邻居事件了。在这里,显示了前面段落里描述的邻居事件的两种情况:路由器Grissom在超出重传次数限制时被宣告丢失,但一旦收到下一个Hello数据包时又立即“找回来”了。

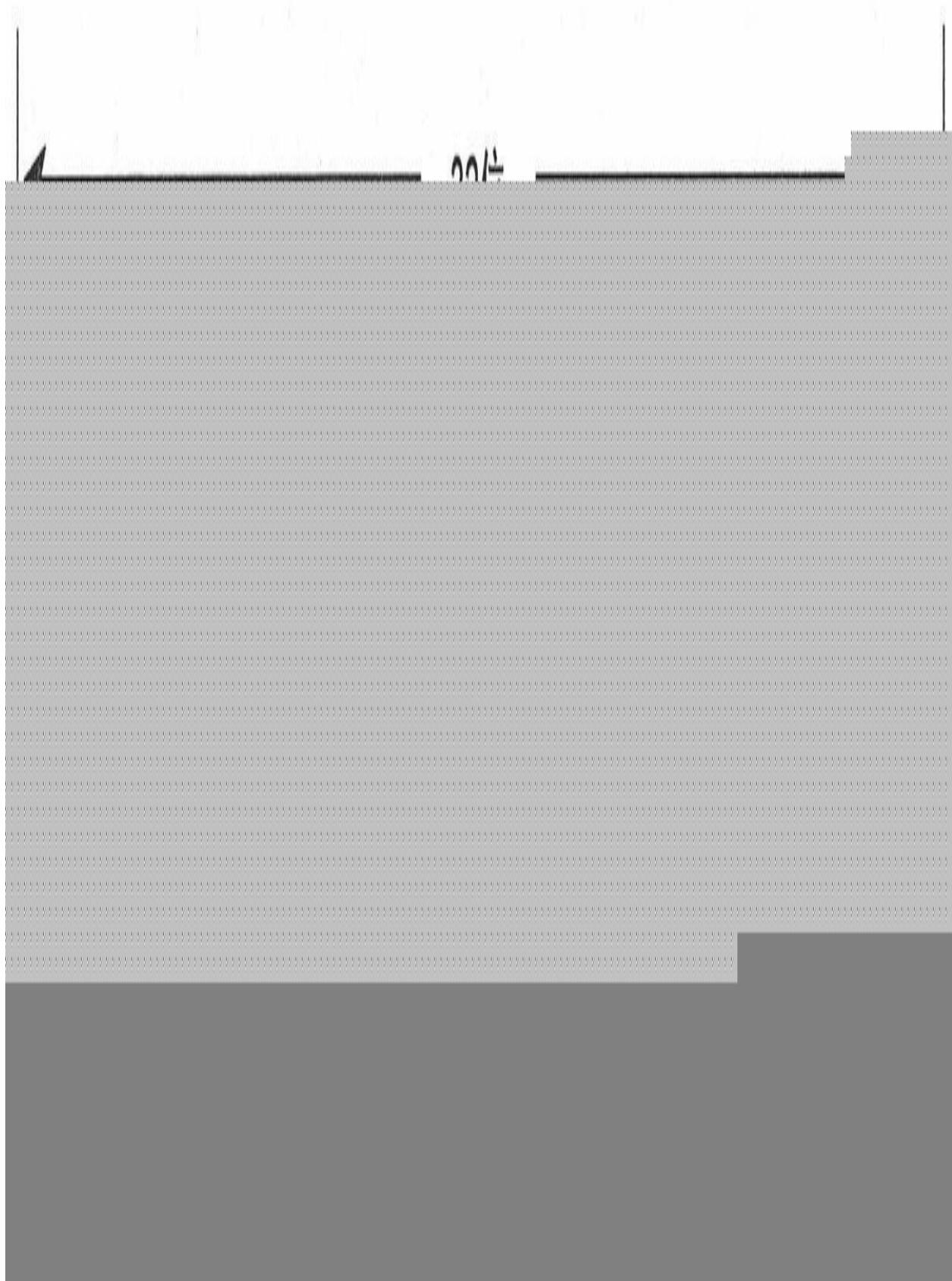
示例7-45 命令**debug eigrp neighbors**显示了邻居事件



虽然示例7-44中显示正在向路由器Grissom发送更新数据包,但是在示

例7-46的路由器Grissom上，通过对EIGRP数据包的观察来看，这台路由器并没有收到那些发送给它的数据包。因为路由器Grissom可以和路由器Cooper成功地交换Hello数据包，因而路由器Grissom的EIGRP进程肯定是在运行的。因此怀疑是路由器Grissom的以太网接口发生故障了。在示例7-47中，观察路由器的配置文件，发现在以太网接口E0处配置了一个作为入站过滤器的访问列表。

示例7-46 路由器Grissom正在和路由器Cooper通过接口S0交换Hello数据包，并正在从接口E0发送出Hello数据包。但是，路由器Grissom却没有在接口E0上收到任何EIGRP的数据包



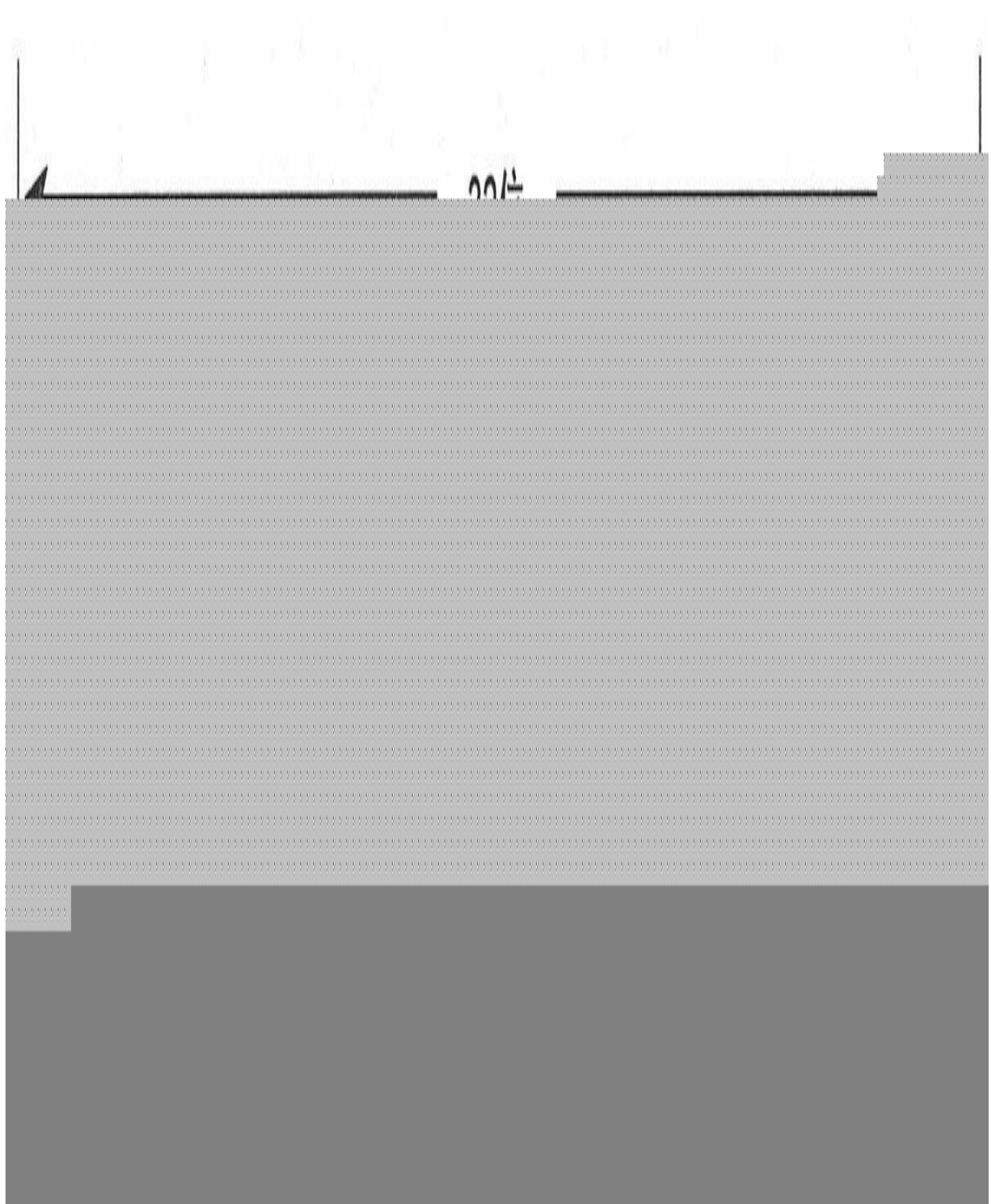
示例7-47 入站访问列表正在拒绝EIGRP数据包



当在路由器Grissom的E0接口上收到EIGRP的数据包时，这些数据包首先要通过访问列表150的过滤。由于这些数据包和访问列表中的任何一项都不匹配，因此它们被丢弃。这个问题可以通过在访问列表后附加一条匹配条目来解决（参见示例7-48）：

```
access-list 150 permit eigrp 192.168.16.16 0.0.0.15 any
```

示例7-48 当在访问列表后增加一个条目来允许EIGRP数据包通过时，路由器Grissom的邻居表和路由表显示出了它现在拥有可达所有子网的路由



7.5.2 “卡”在活动状态的邻居

当本地路由器的一条路由变为活动状态并且向它的邻居路由器发送查询

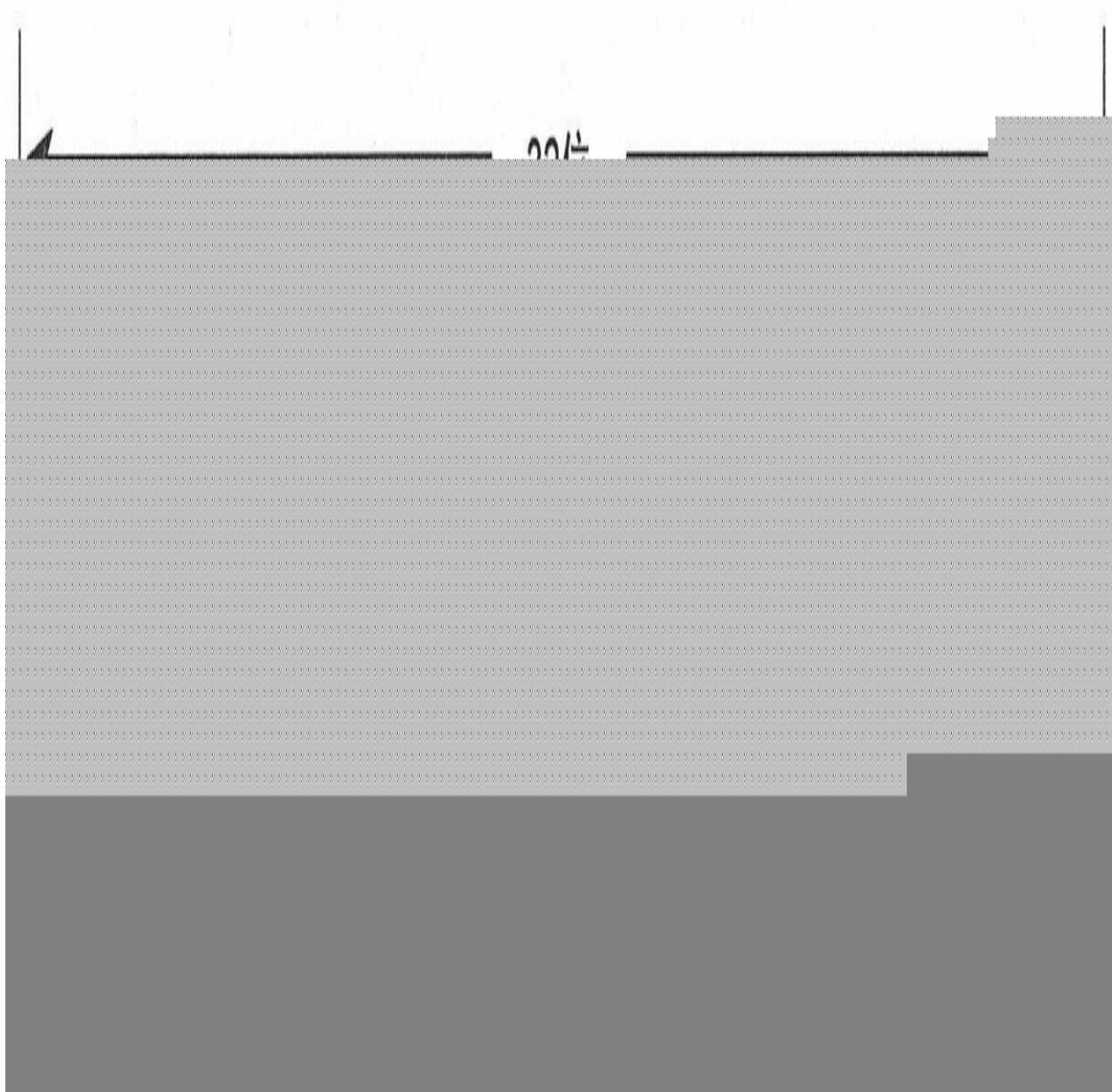
时，在本地路由器收到每个查询的答复之前，这条路由将一直保持活动状态。但是如果一个邻居失效或其他无效的情况发生将没有办法作出答复，那么究竟会发生什么呢？答案是这条路由将会永久地停留在活动状态。活动计时器和SIA-Retransmit计时器将设计用来防止这种情况的发生。当发送一个查询时，就会设置活动计时器和SIA-Retransmit计时器。如果路由器的IOS软件（IOS版本早于12.2[4.1]）不支持SIA-Retransmit计时器，则只能使用活动计时器。如果在收到查询的答复消息之前，活动计时器超时了，这条路由就被宣告“卡”在活动状态（stuck-in-active），这个邻居也就被推断为失效，并且从邻居表中刷新掉。[\[21\]](#) SIA路由和任何其他经过这个邻居的路由也都会从路由表中删除。DUAL算法将会认为这个邻居已经答复了一个含有无穷大度量的消息。

事实上，这一系列的事件应该从来不会发生。在活动计时器超时很久以前，丢失的Hello 数据包就应该识别出一个无效的邻居了。

但是在一个大型的EIGRP网络中，究竟发生了什么，使一个查询消息就像电池广告中的小兔子一样，一直保持向前继续？回忆一下，查询会引起扩散计算的不断增大，反之，答复将会引起扩散计算的不断减小，如图7-6所示。这样，查询最终将必然会到达网络的边界，而答复最终也必然开始往回收缩，但是，如果扩散计算的直径增大到足够大，活动计时器将可能在收到所有的答复前超时。结果，从邻居表中刷新掉一个合法的邻居将明显会带来网络的不稳定。

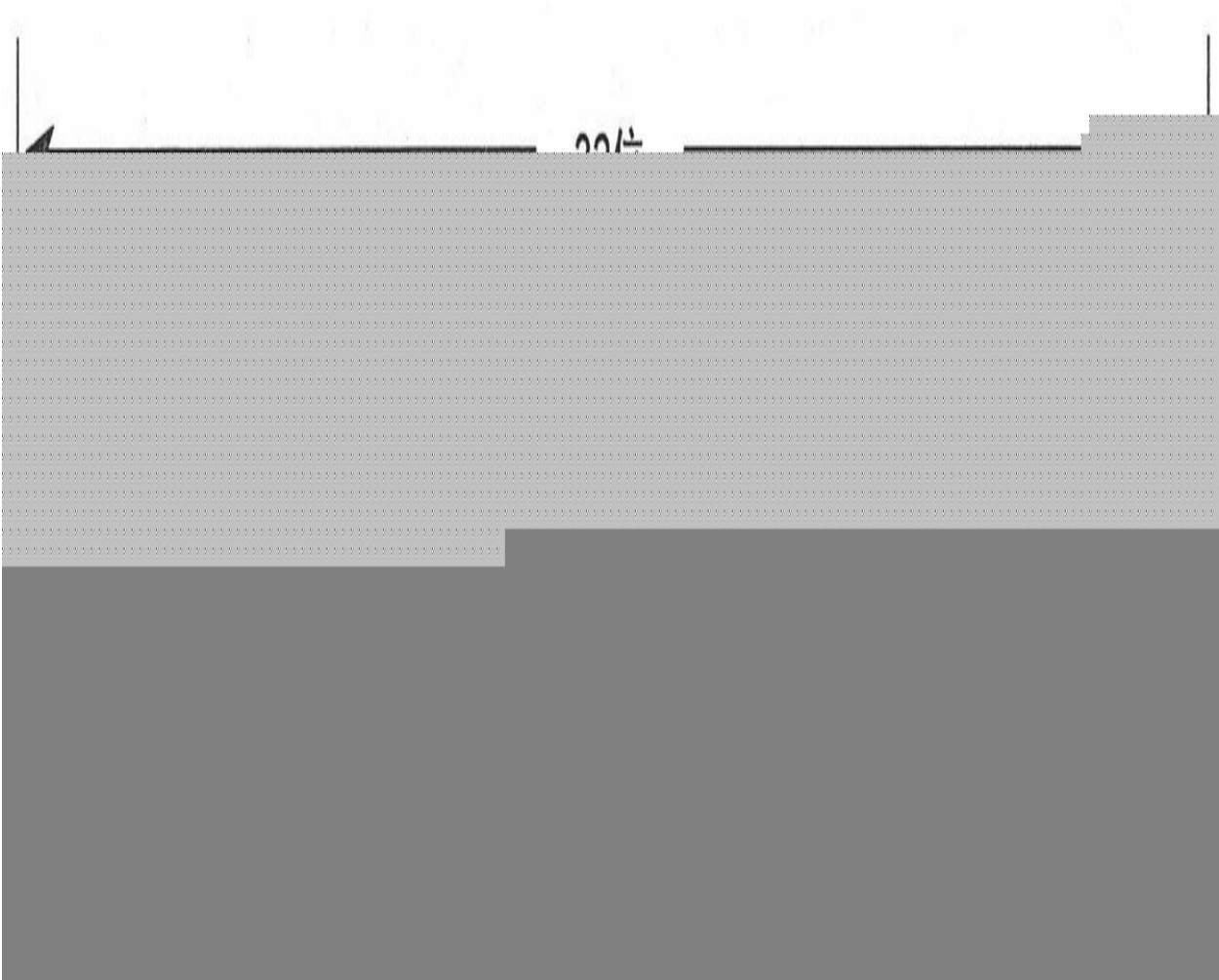
当一个邻居神秘地从邻居表中消失后，随后又重新出现，或者用户抱怨总是断断续续地不能到达目的地时，SIA路由可能就是故障所在了。检查路由器的错误记录日志是找出是否出现SIA路由的一个好途径，参见示例7-49所示。

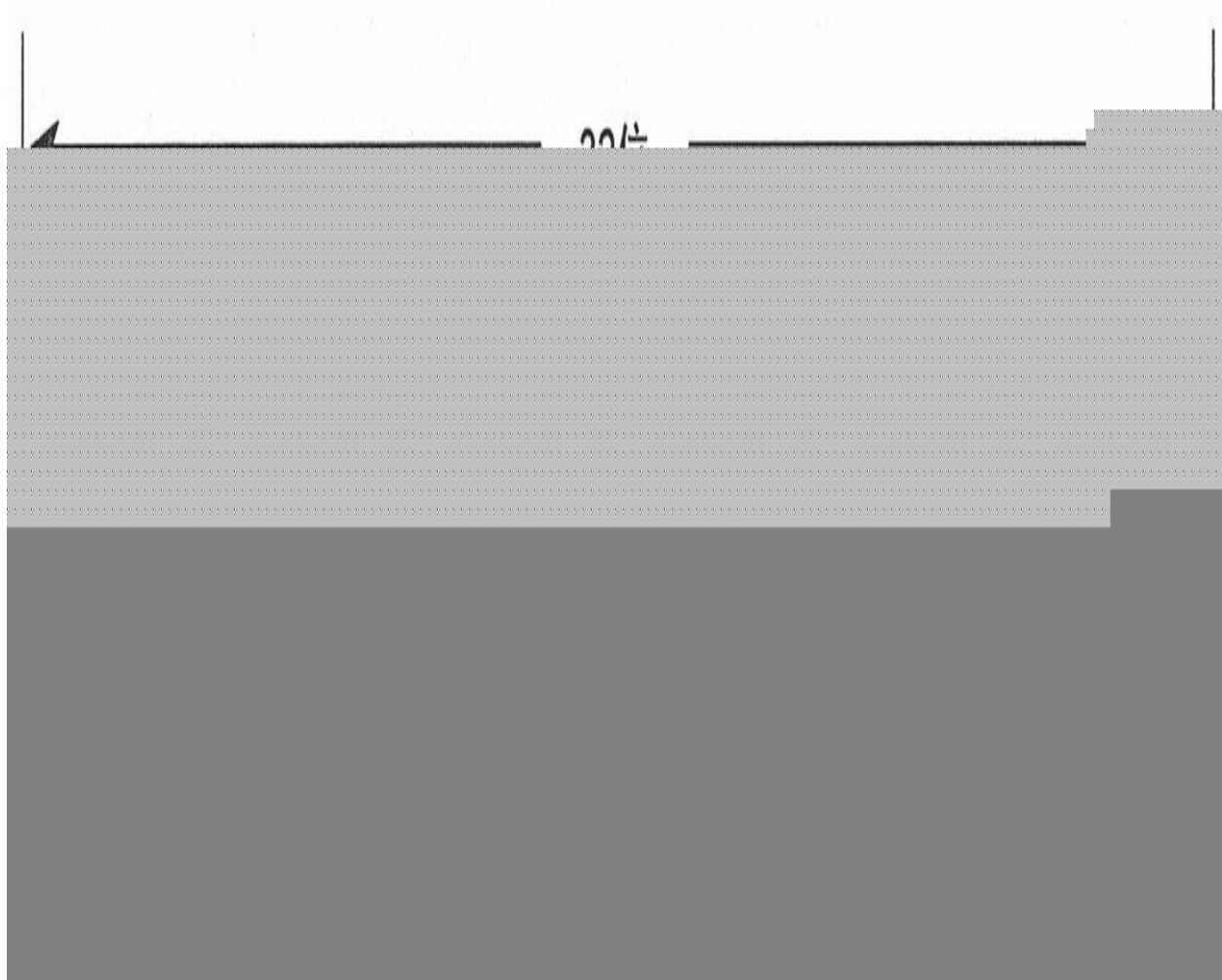
示例7-49 这个错误记录日志中的最后一条记录显示了一条SIA消息



当追踪SIA路由产生的原因时，应该仔细关注路由器的拓扑结构表。如果路由处于活动状态，那么就应该注意邻居路由器仍然没有收到查询内容。例如，在示例7-50中显示了一个含有几条处于活动状态的路由的拓扑结构表。注意，这里大多数的路由已经有15s的时间处于活动状态了，而另一个路由（10.6.1.0）进入活动状态则已经有41s的时间了。

示例7-50 这个拓扑结构表显示了几个处于活动状态的路由，它们都在等待来自邻居路由器**10.1.2.1**的答复





也注意到在每个条目中，邻居路由器10.1.2.1都带有一个答复状态标记（r），这表明该邻居的答复仍然没有被收到。邻居路由器本身或者和邻居路由器相连的链路可能没有问题，但是在网络拓扑中该信息指出了应该将追查继续下去的方向。

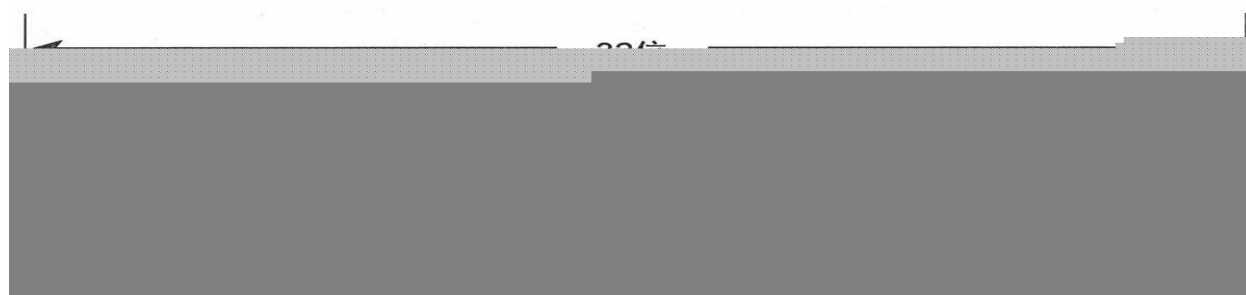
通常在一个大型的EIGRP网络中引起SIA的原因是网络拥塞严重、数据链路带宽较低以及路由器内存过低或CPU利用率负荷过大，等等。如果这些有限的资源必须处理数量很大的查询的话，这个问题将会进一步恶化。

冒然地调整接口上的带宽参数可能会引起另外的SIA路由。回忆一下在设计EIGRP网络时，EIGRP使用的带宽一般不超过链路可用带宽的50%。这个限制意味着EIGRP的调节是和所配置的带宽相关联的。如果试图在处理路由选择时人为地降低带宽，EIGRP进程所需求的带宽将可

能会极度缺乏。假如运行的是IOS软件的11.2版本或后续的版本，则可以使用命令**ip bandwidth-percent eigrp** 来调整带宽使用的百分比。

例如，假设一个接口和一个带宽为56kbit/s的串行链路相连，但是链路带宽被设置成了14kbit/s。EIGRP协议应该限制自己使用的带宽应在这个数值的50%之内，或者说就是在7kbit/s之内。示例7-51中的命令将把EIGRP协议使用的带宽百分比调整到200%，即14kbit/s的200%，也就是实际56kbit/s链路带宽的50%：

示例7-51 路由器的配置调整了**EIGRP**占用的配置带宽的百分比



也可以使用命令**timers active-time** 来增大活动计时器的周期，这样在某些情况下可以帮助避免SIA路由，但是采取这种办法应当仔细考虑对网络路由再次收敛的影响。

SIA-Retransmit计时器是一个新的计时器，它和两个新的EIGRP数据包类型——SIA查询和SIA答复一起帮助使SIA减少到最少，并对响应查询确实有问题的链路上的邻居进行重置。

考虑图7-39中的网络。对于路由器Mercury, EIGRP路由会引导到达网络172.16.100.0的流量经过路由器Apollo。路由器Vostok不是一台可行后继路由器，因为从路由器Vostok到网络172.16.100.0的度量值太大。路由器Vostok到达172.16.100.0的流量会引导经过路由器Mercury和Apollo。路由器Soyuz不是一台可行后继路由器，因为从Soyuz到达网络172.16.100.0的度量值太大了。

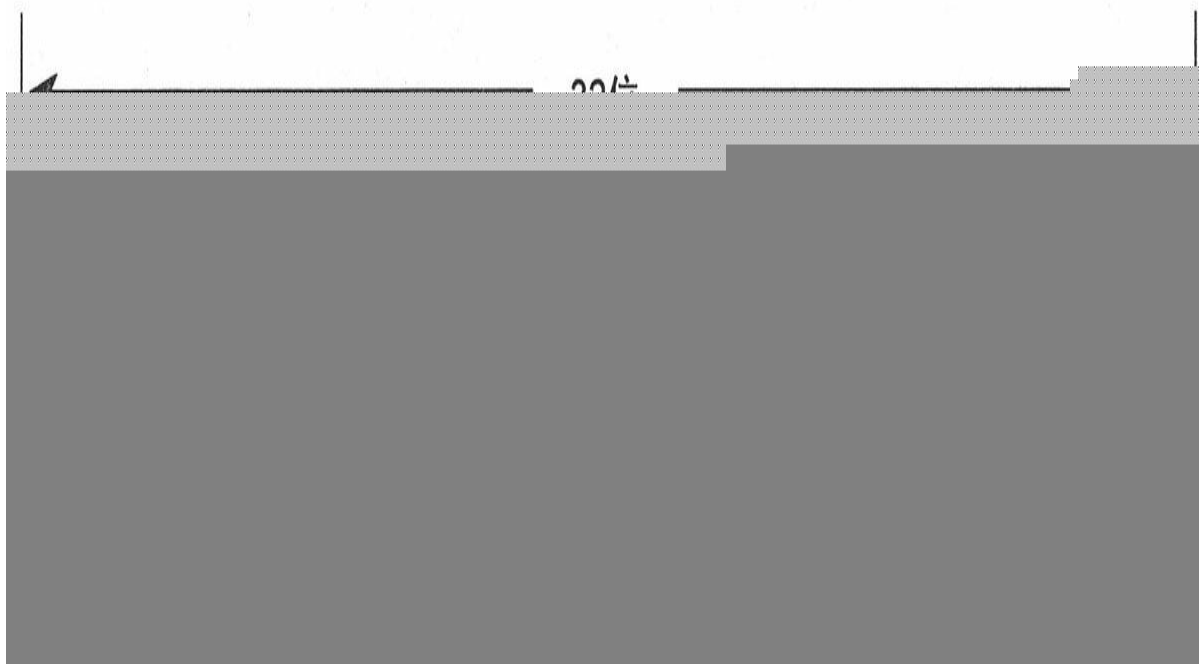


图7-39 路由器**Mercury**并不把路由器**Vostok**作为到达网络**172.16.100.0**的一台可行后继路由器

如图7-40所示，当路由器**Mercury**和**Apollo**之间的链路发生故障时，路由器**Mercury**会把地址**172.16.100.0**（和经过邻居**Apollo**学习到的任何其他地址）变为活动状态，并发送一个查询到路由器**Vostok**。路由器**Vostok**也会把该地址置为活动状态，并发送一个查询到路由器**Soyuz**。这时活动计时器将被设置。另外，**SIA-Retransmit**计时器也会被设置。**SIA-Retransmit**计时器设置成活动计时器的数值的一半，通常是90s。

在**SIA-Retransmit**计时器超时以后，路由器**Mercury**发送一个**SIA**查询到路由器**Vostok**。路由器**Vostok**发送一个**SIA**答复到路由器**Soyuz**。路由器**Vostok**将会发送一个**SIA**答复响应来自路由器**Mercury**的**SIA**查询。路由器**Mercury**会重置活动计时器和**SIA-Retransmit**计时器。在接收**SIA**答复消息时这些路由器会发送最多3个**SIA**查询（假定从始发地址查询的地方没有收到答复消息），然后才会重置一台邻居路由器。因此在一台邻居路由器响应**SIA**查询的时候（大约6min，假定缺省的活动计时器为180s），它不应该被宣告为stuck-in-active和重置。这对于在一个大型网络中对查询作出响应已经给了足够的时间了。

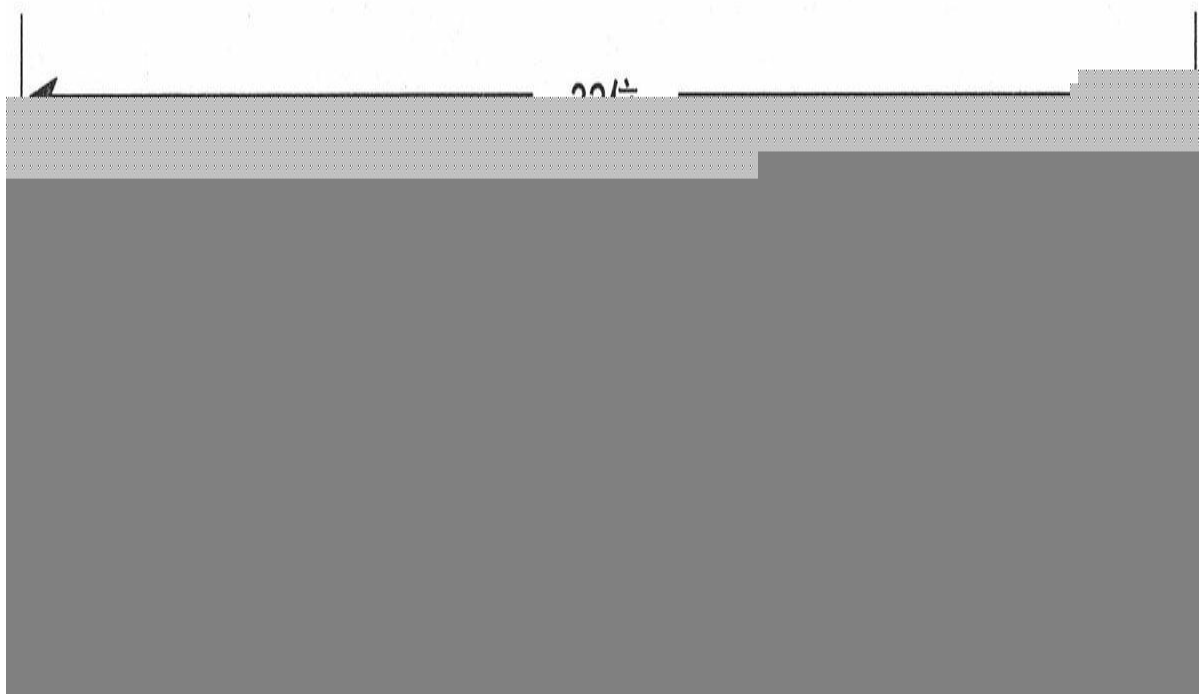


图7-40 SIA查询和SIA答复用来避免SIA情况

但是，在路由器Vostok到Soyuz之间的链路上会存在一个问题，虽然它允许使用足够多的Hello数据包用来保持邻居路由器的活动状态，但是在SIA-Retransmit的计时周期内还没有收到来自路由器Vostok的SIA答复。如果在SIA查询的90s计时时间内没有收到SIA答复，从而对始发地址查询的响应也不会收到，路由器Vostok将会重置邻居路由器Soyuz，并答复路由器Mercury始发的查询宣告那个地址不可到达。

SIA-Retransmit计时器可以完成两件事情。如果邻居路由器正在对SIA查询进行响应，那么将会给大型网络更多的时间来对地址的查询作出响应。如果邻居路由器没有进行响应，那么邻居路由器将被重置。只有从它的邻居没有收到响应的路由器才会重置它的邻接关系。在引入SIA-Retransmit计时器之前，即使网络下游的某处出现了问题，在活动计时器超时后如果还没有收到对一个活动查询的响应，那么路由器将会重置它的邻居邻接关系。

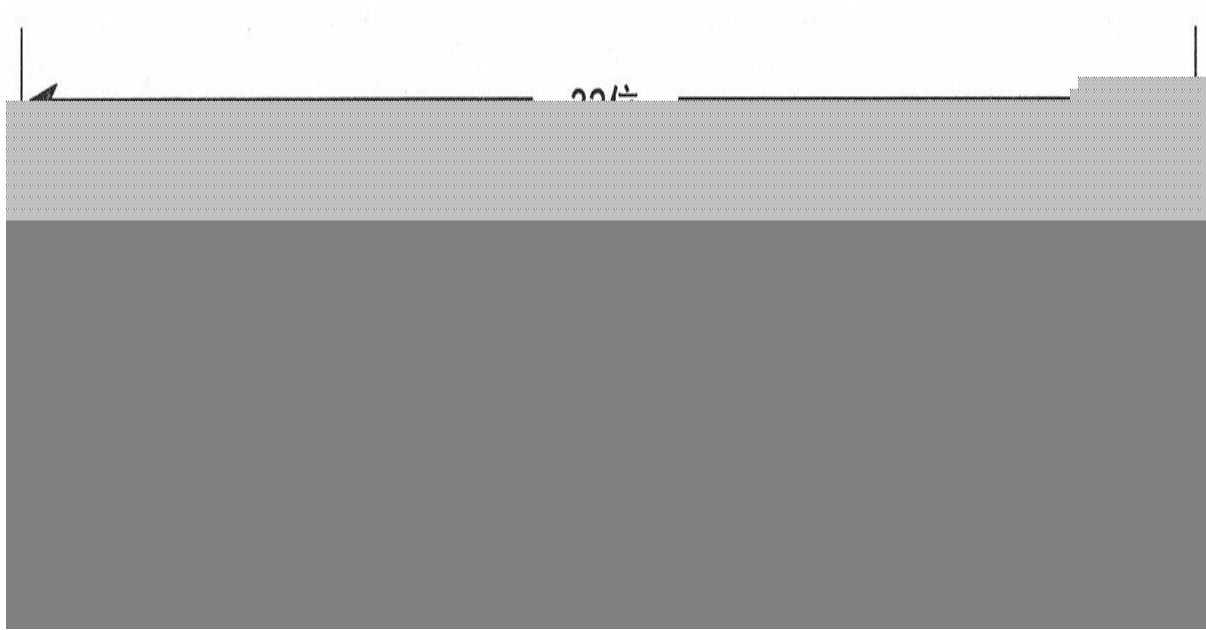
一个好的网络设计应该是解决像SIA路由这类网络不稳定性的最好方法。全面考虑使用灵活的地址分配、路由过滤、缺省路由和路由汇总，在一个大型的网络中创建一些边界来限制扩散计算的大小和范围。第13章将包含一个这种网络设计的例子。

7.6 展 望

当比较EIGRP协议和OSPF协议时，人们经常说EIGRP协议的优势在于它的配置比较简单。这种看法在很多网络的情况中是正确的，但是本章故障处理一节的讲述显示了当网络的规模增大时，对划分EIGRP网络的拓扑将做出更多地努力。相反地，正如下一章所描述的，非常复杂的OSPF在配置一个大型网络时反而显得简单了。

7.7 总结表：第7章命令总结

2012



7.8 复习题

1. EIGRP协议是一个距离矢量协议还是一个链路状态路由选择协议？
2. EIGRP协议在一条链路上使用的可配置最大带宽是多少？这个带宽百分比可以更改吗？
3. EIGRP协议和IGRP协议在计算复合度量时有什么不同？
4. EIGRP协议的4个基本部件是什么？
5. 在EIGRP协议的上下文环境中，术语“可靠的分发（reliable delivery）”是什么意思？有哪两种方法可以确保EIGRP数据包的可靠分发？
6. 有什么机制可以确保路由器正在接收的是最新的路由条目？
7. EIGRP协议使用的组播IP地址是什么？
8. EIGRP协议使用的数据包类型是什么？
9. 缺省情况下，EIGRP协议发送Hello数据包的时间间隔是多少？
10. 缺省的抑制时间是多少？
11. 邻居表和拓扑结构表的不同之处是什么？
12. 什么是可行距离？
13. 什么是可行性条件？
14. 什么是可行后继路由器？
15. 什么是后继路由器？
16. 处于活动状态的路由和处于被动状态的路由有什么不同之处？
17. 引起一个被动状态的路由变成活动状态的条件是什么？

18. 引起一个活动状态的路由变成被动状态的条件是什么？
19. Stuck-in-active是什么意思？
20. 子网划分和地址聚合有什么不同之处？

7.9 配置练习

1. 写出图7-41中路由器A、B和C的EIGRP配置，并使用进程ID 5。



图7-41 配置练习1和练习2的网络

2. 在图7-41中，连接路由器A和路由器B的串行接口都是S0接口。配置这两台路由器之间的认证，假设从今天开始，有两天的时间使用第一个钥匙。然后配置第二个钥匙，在第一个钥匙使用过后30天开始使用。

在图7-42中增添了路由器D。将这台路由器添加到配置练习2所写的配置中。

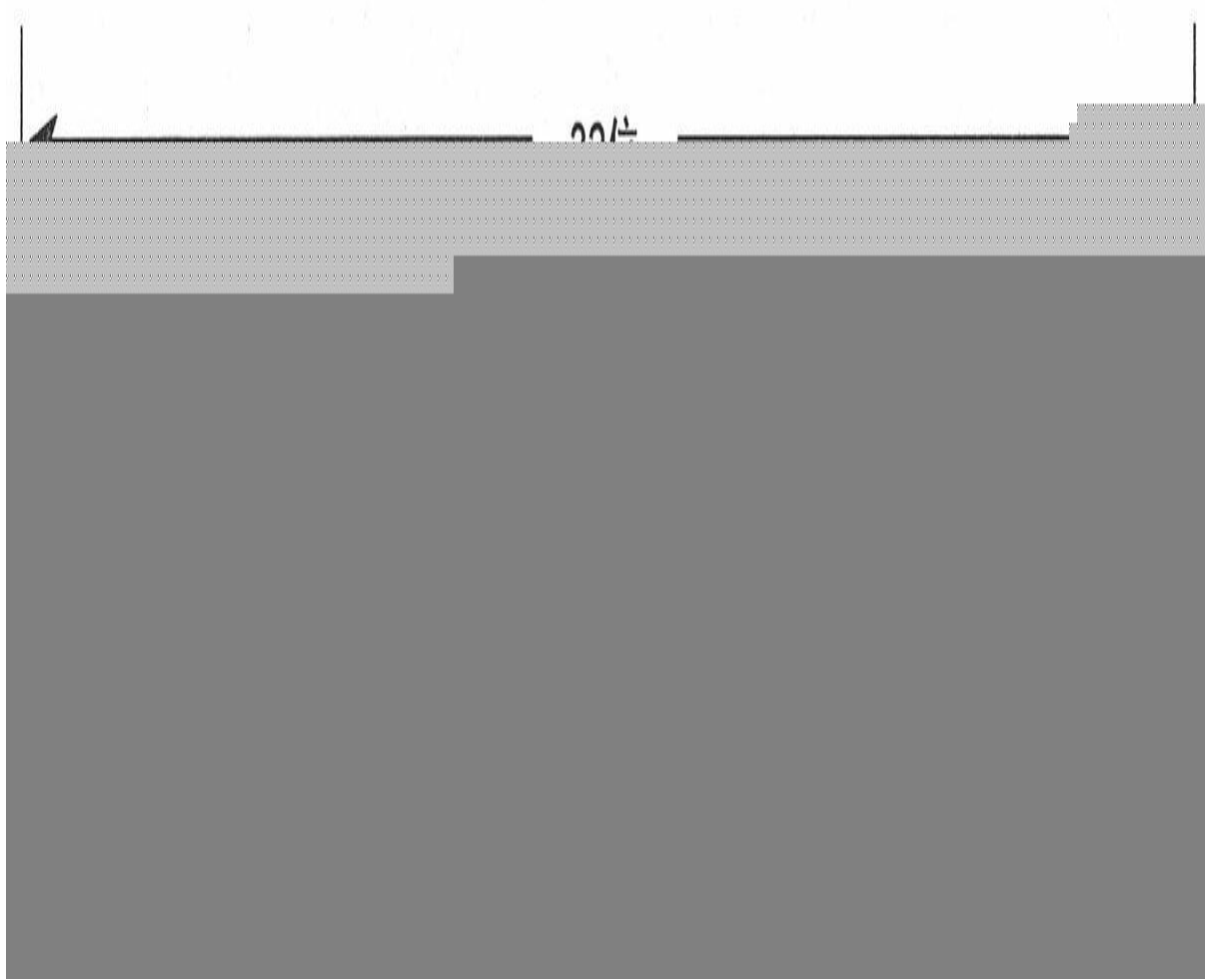


图7-42 配置练习3的网络

3. 在图7-43中增添了路由器F，配置这台路由器，在与配置练习2和练习3中配置的路由器之间运行EIGRP协议。
4. 在图7-43的网络中任何可能的地方配置路由汇总。

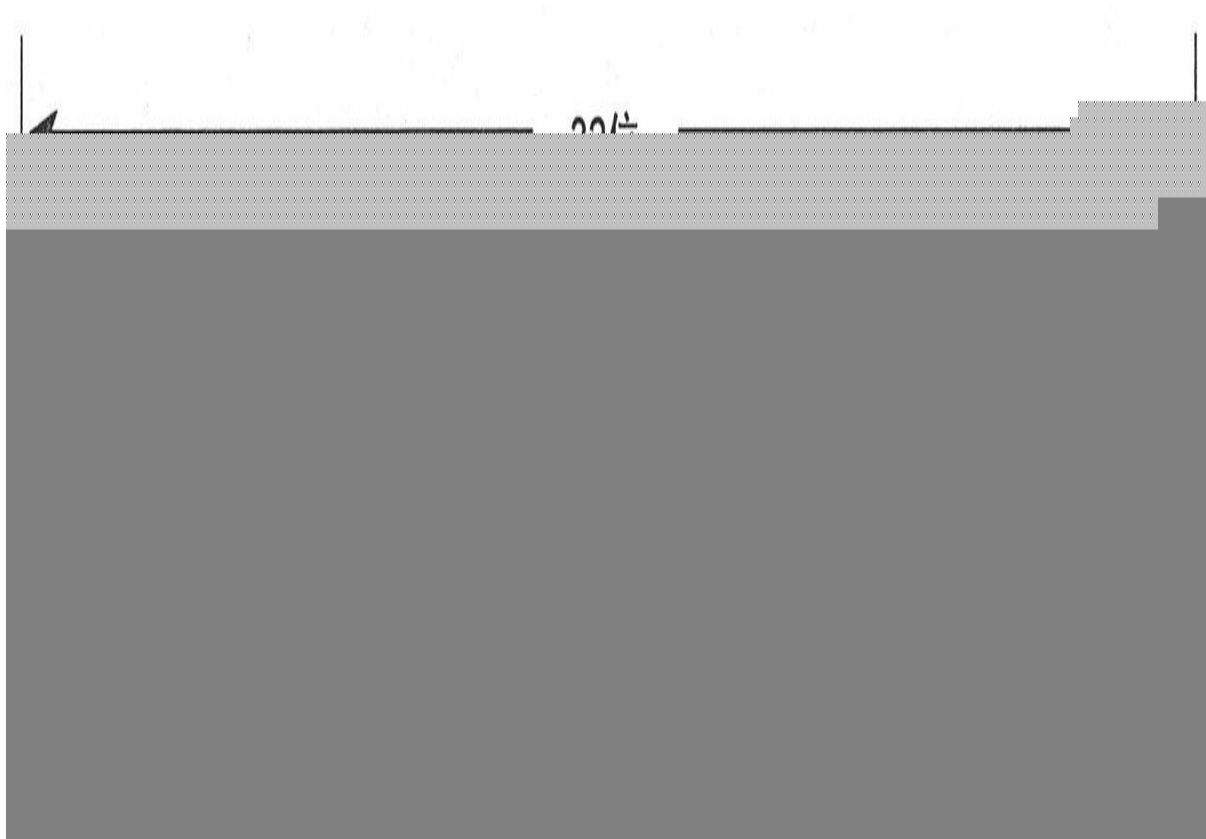


图7-43 配置练习4和练习5的网络

7.10 故障排除练习

1. 表7-7显示了在图7-44中的每个接口上使用**show interface** 命令显示的数值。哪一台路由器将作为路由器F到达子网A的后继路由器？

表7-7 使用**show interface**命令显示的图7-44中有接口的度量值

4 2015

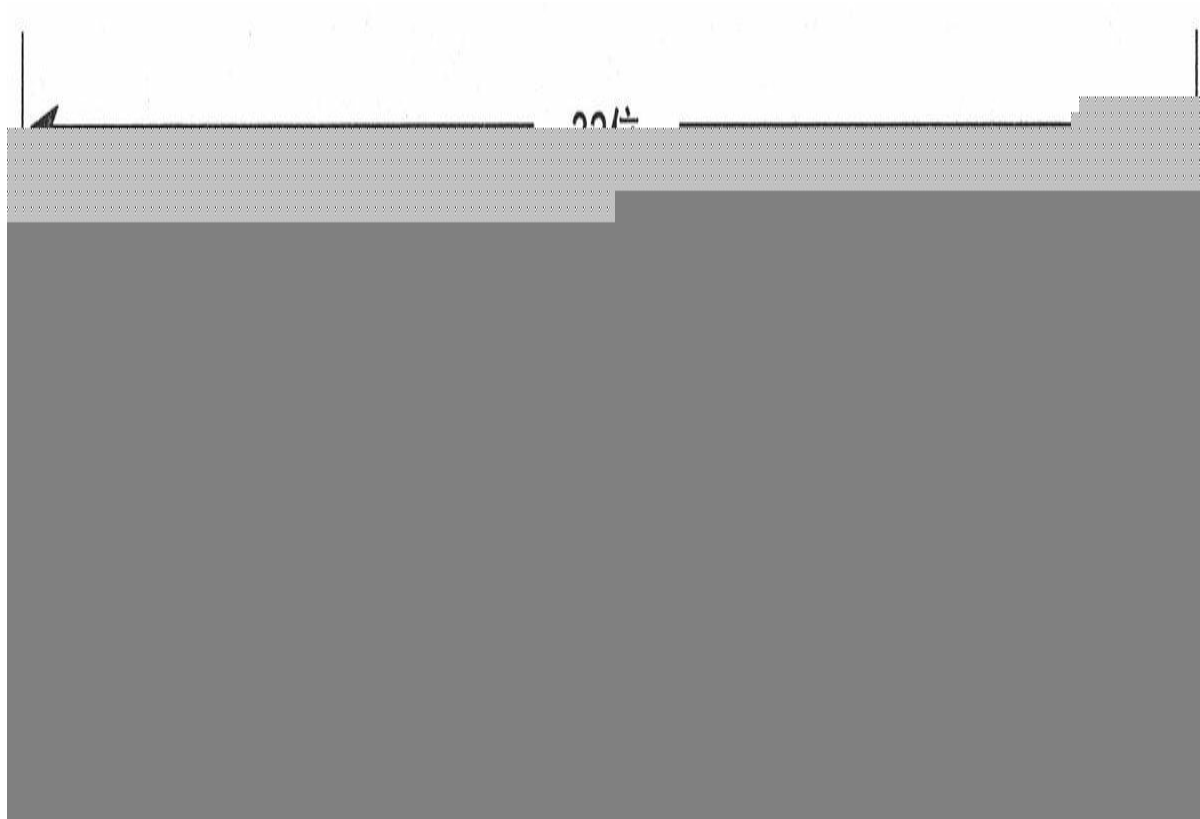
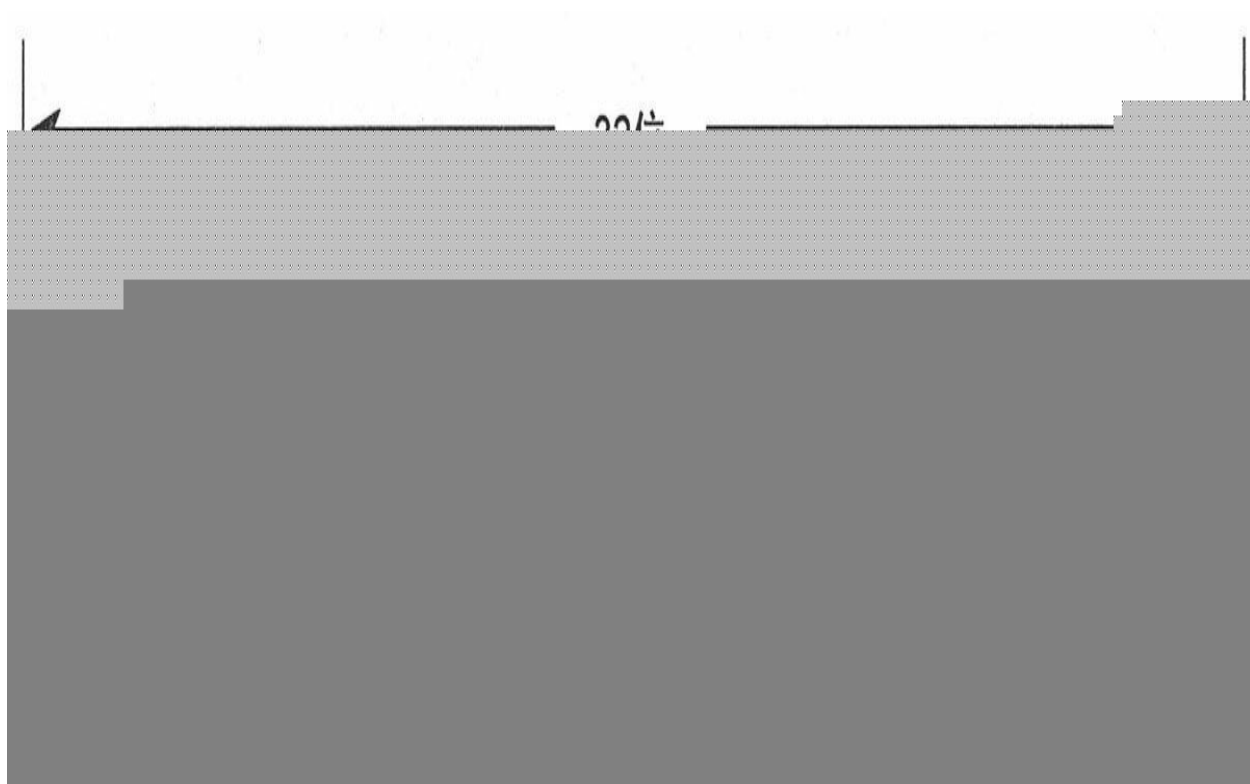


图7-44 故障排除练习1~5的网络

2. 在图7-44中，路由器C到达子网A的可行距离是什么？
 3. 在图7-44中，路由器G到达子网A的可行距离是什么？
 4. 在图7-44中，在路由器G的拓扑结构表中，有哪些路由器是作为可行后继路由器的？
 5. 在图7-44中，路由器A到达子网B的可行距离是什么？
-

[1] 这个完整性检查可以用命令no validate-update-sourced来禁止它生效。

[2] 把缺省网络分类归到外部路由是IGRP和EIGRP协议所独有的。像RIP和OSPF等开放性的协议使用地址0.0.0.0通告缺省网络。

[3] tos是Cisco公司最早打算使用IGRP协议来提供支持服务类型的路由选择时遗留下来的参数；但这个计划从来没有被采纳过，因而tos的值总是被设定为0。

[4] 译者注：原来的度量计算公式将变为 $\text{metric} = k1 \times BW_{\text{IGRP}(\min)} + (k2 \times BW_{\text{IGRP}(\min)} / (256 - \text{LOAD}) + k3 \times DLY_{\text{IGRP}(\text{sum})}$ 。

[5] 需要注意，IGRP协议的管理距离是100。

[6] 点到点的子接口每5s发送一次Hello消息。

[7] Edsger W. Dijkstra和C. S. Scholten编写的“Termination Detection for Diffusing Computations.”Information Processing Letters, Vol.11, No. 1,pp. 1-4:29 August 1980。

[8] J. J. Garcia-Luna-Aceves.“A Unified Approach for Loop-Free Routing Using Link States or Distance Vectors,” ACM SIGCOMM Computer Communications Review, Vol. 19,No.4,pp.212-223: September 1989。
J.J.Garcia-Luna-Aceves. “Loop-Free Routing Using Diffusing Computations,” IEEE//ACM Transactions on Networking, Vol.1,No.1,February 1993。

[9] J.J. Garcia-Luna-Aceves. “Area-Based, Loop-Free Internet Routing.” Proceedings of IEEE INFOCOMM 94. Toronto, Ontario, Canada, June 1994.

[10] 后继路由器简单的理解就是指到达目的网络更近一跳的路由器，换句话说，就是下一跳路由器。

[11] 事实上，这个接口并未明确地显示在路由表中。更确切地说，它是邻居路由器自身的属性。这个约定意味着通过多条并行链路的相同路由器将被EIGRP协议看作是多个邻居。

[12] 在本小节和后续几小节演示的几个图例以及使用的网络示例改编自Garcia-Luna先生的“Loop-Free Routing Using Diffusing Computations”，并得到了他的许可。

[13] 译者注：在一些早期的IOS软件版本中，缺省的活动计时器设置为1min。

[14] 无穷大距离可以用OxFFFFFFFF或4294967295来表示。

[15] 更正确地说，聚合应该是任何一组地址的汇总。这里声明，本书中所提及的聚合地址是指一组主网络地址的汇总。

[16] V. Fuller, T. Li, J. I. Yu, and K. Varadhan. “Classless Inter-Domain Routing (CIDR) :An Address Assignment and Aggregation Strategy.” RFC 1519, 1993年9月。

[17] 记住串行接口的缺省带宽是1544kbit/s。

[18] 缺省的路径是4条，可以参见7.4.3小节来获取进一步的详细信息。

[19] 虽然MD5认证是EIGRP协议目前惟一可用的认证模式，但是使用命令ip authentication mode eigrp md5可以为将来出现其他可用的认证模式作预先考虑。

[20] 当排除一个网络的故障时，检查所有路由器接口的地址配置是否属于正确的子网是一种好习惯。

[\[21\]](#) 正如前面所提及的，缺省的活动计时时间是3min，并且可以通过命令timer active-time来更改。

本章包括以下主题：

- OSPF协议的基本原理与实现；
- OSPF协议的配置；
- OSPF协议的故障诊断。

第8章

开放最短路径优先 协议（**OSPFv2**）

开放最短路径优先（Open Shortest Path First, OSPF）协议是由Internet工程任务组（Internet Engineering Task Force, IETF）开发的路由选择协议，用来替代存在一些问题的RIP协议。现在，OSPF协议是IETF组织建议使用的内部网关协议（IGP）。OSPF协议是一个链路状态协议，正如它的命名所描述的，OSPF使用Dijkstra的最短路径优先（SPF）算法，而且是开放的。这里所说的开放是指它不属于任何一个厂商或组织所私有。OSPF协议的发展经过了几个RFC，所有这些相关的RFC都是由John Moy撰写的。RFC 1131详细说明了OSPF协议版本1，这个版本从来没有在实验平台以外使用过。OSPF协议版本2，也就是目前IPv4协议仍然使用的版本，最初是在RFC 1247中说明的，最新是在RFC 2328中说明的。

像所有的链路状态协议一样，OSPF协议和距离矢量协议相比，一个主要的改善在于它的快速收敛，这使得OSPF协议可以支持更大型的网络，并且不容易受到有害路由选择信息的影响。OSPF协议的其他一些特性有：

- 使用了区域的概念，这样可以有效地减少路由选择协议对路由器的CPU和内存的占用；划分区域还可以降低路由选择协议的通信量，这使构建一个层次化的网络拓扑成为可能；
- 完全无类别地处理地址问题，排除了不连续子网这样的有类别路由选择协议的问题；
- 支持无类别路由表查询、VLSM和用来进行有效地址管理的超网技术；
- 支持无大小限制的、任意的度量值；

- 支持使用多条路径的效率更高的等价负载均衡；[\[1\]](#)
- 使用保留的组播地址来减小对不宣告OSPF的设备的影响；
- 支持更安全的路由选择认证；
- 使用可以跟踪外部路由的路由标记。

OSPF协议也支持具有服务类型（Type of Service, ToS）的路由选择能力，但是它从来没有被广泛地实施过。基于这个原因，RFC 2328已经在OSPF协议中删除了该ToS路由选择选项。

8.1 OSPF的基本原理与实现 [2]

从一个非常概括的角度来看，OSPF协议的操作是比较容易解释的：

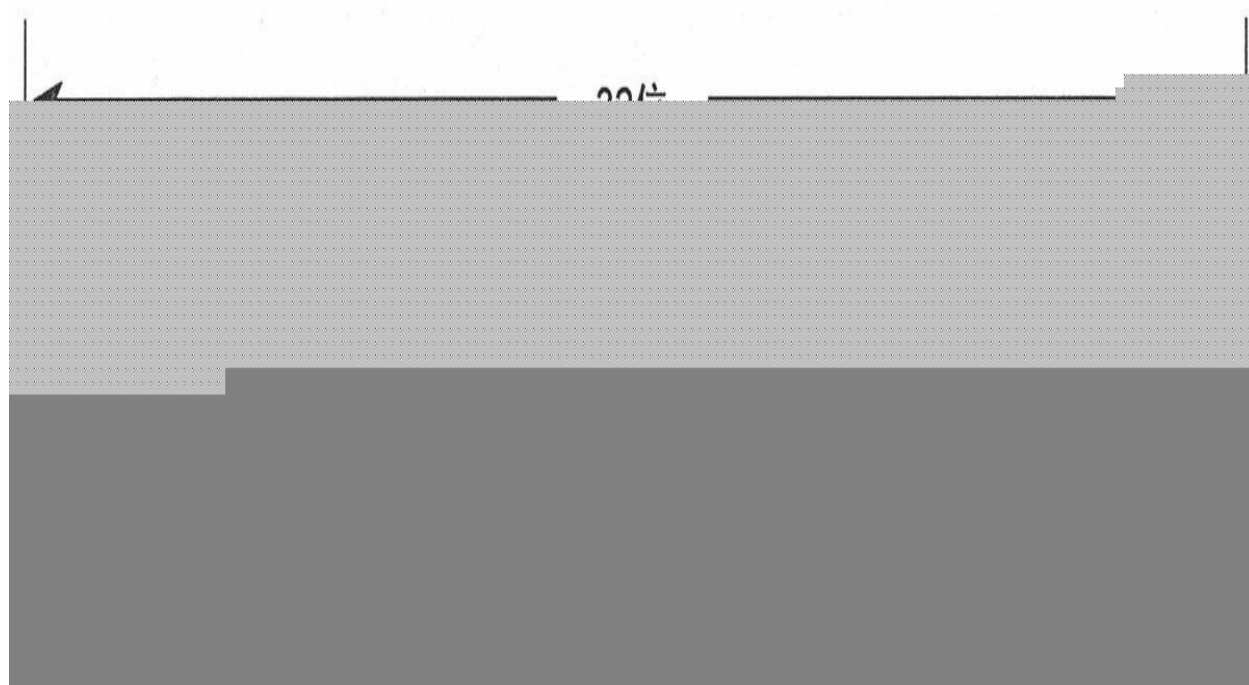
1. 宣告OSPF的路由器从所有启动OSPF协议的接口上发出Hello数据包。如果两台路由器共享一条公共数据链路，并且能够相互成功协商它们各自Hello数据包中所指定的某些参数，那么它们就成为了邻居（Neighbor）。
2. 邻接关系（Adjacency），可以想象成为一条点到点的虚链路，它是在一些邻居路由器之间构成的。OSPF协议定义了一些网络类型和一些路由器类型的邻接关系。邻接关系的建立是由交换Hello信息的路由器类型和交换Hello信息的网络类型决定的。
3. 每一台路由器都会在所有形成邻接关系的邻居之间发送链路状态通告（Link State Advertisement, LSA）。LSA描述了路由器所有的链路、接口、路由器的邻居以及链路状态信息。这些链路可以是到一个末梢网络（stub network，是指没有和其他路由器相连的网络）的链路、到其他OSPF路由器的链路、到其他区域网络的链路，或是到外部网络（从其他的路由选择进程学习到的网络）的链路。由于这些链路状态信息的多样性，OSPF协议定义了许多LSA类型。
4. 每一台收到从邻居路由器发出的LSA的路由器都会把这些LSA记录在它的链路状态数据库当中，并且发送一份LSA的拷贝给该路由器的其他所有邻居。
5. 通过LSA泛洪扩散到整个区域，所有的路由器都会形成同样的链路状态数据库。
6. 当这些路由器的数据库完全相同时，每一台路由器都将以其自身为根，使用SPF算法来计算一个无环路的拓扑图，以描述它所知道的到达每一个目的地的最短路径（最小的路径代价）。这个拓扑图就是SPF算法树。
7. 每一台路由器都将从SPF算法树中构建出自己的路由表。 [3]

当所有的链路状态信息洪泛到区域内的所有路由器上，并且邻居检验它们的数据库也相同（也就是说，链路状态数据库已经同步），从而成功创建路由表时，OSPF协议就变成了一个“安静”的协议。邻居之间交换的Hello数据包称为keepalive，并且每隔30min重传一次LSA。如果网络拓扑稳定，那么网络中将不会有什么活动或行为发生。

8.1.1 邻居和邻接关系

在发送任何LSA通告之前，OSPF路由器都必须首先发现它们的邻居路由器并建立起邻接关系。邻居路由器，连同每一台邻居路由器所在的链路（接口）和维护邻居路由器的一些必要的其他信息都被记录在一个邻居表里，参见示例8-1所示。

示例8-1 邻居表记录了所有宣告OSPF协议的邻居路由器



一台OSPF路由器对其他OSPF路由器的跟踪需要每台路由器都提供一个路由器ID（Router ID），路由器ID在OSPF区域内惟一标识一台路由器的IP地址。Cisco路由器通过下面的方法得到它们的路由器ID：

- （1）如果使用**router-id** 命令手工配置Router ID，就使用Router ID。

(2) 如果没有手工配置Router ID，路由器就选取它所有的四环(loopback)接口上数值最高的IP地址。

(3) 如果路由器没有配置IP地址的loopback接口，那么路由器将选取它所有的物理接口上数值最高的IP地址。用作路由器ID的接口不一定非要运行OSPF协议。

使用loopback接口作为路由器ID有两个好处：

- loopback接口比任何其他物理接口更稳定。一旦路由器启动成功，这个环回接口就处于活动状态，只有整个路由器失效时它才会失效。
- 网络管理员在预先分配和识别作为路由器ID的地址时有更多的回旋余地。

在Cisco路由器上，即使路由器的这个用作路由器ID的物理接口随后失效或被删除，OSPF协议也会继续使用原来的物理接口作为路由器ID（参见本章后面的8.2.2小节的内容）。因此，loopback接口的稳定性只是一个次要的优点，loopback接口的一个主要好处在于它具有更好控制路由器ID的能力。

OSPF路由器利用Hello数据包通告它的路由器ID来开始建立和邻居的关系。

1. Hello协议

Hello协议服务于以下几个目的：

- 它是发现邻居路由器的方法；
- 在两台路由器成为邻居之前，需要通告这两台路由器必须相互认可的几个参数；
- Hello数据包在邻居路由器之间担当keepalive的角色；
- 它确保了邻居路由器之间的双向通信；

- 它用来在一个广播网络或非广播多路访问（NBMA）网络上选取指定路由器（Designated Router, DR）和备份指定路由器（Backup Designated Router, BDR）。

宣告OSPF的路由器周期性地从启动OSPF协议的每一个接口上发送Hello数据包。该周期性的时间段称为Hello时间间隔（HelloInterval），它的配置是基于路由器的每一个接口的。在Cisco路由器上，对于广播型网络使用的缺省Hello时间间隔是10s，对于非广播型网络缺省是30s。这个值可以通过命令**ip ospf hello-interval** 来更改。如果一台路由器在一个称为路由器无效时间间隔（RouterDeadInterval）的时间段内还没有收到来自邻居的Hello数据包，那么它将宣告它的邻居路由器无效。在Cisco路由器中，路由器无效时间间隔的缺省值是Hello时间间隔的4倍，并且这个值可以通过命令**ip ospf dead-interval** 来更改。[\[4\]](#)

每一个Hello数据包都包含以下信息：

- 始发路由器的路由器ID（RouterID）；
- 始发路由器接口的区域ID（AreaID）；
- 始发路由器接口的地址掩码；
- 始发路由器接口的认证类型和认证信息；
- 始发路由器接口的Hello时间间隔；
- 始发路由器接口的路由器无效时间间隔；
- 路由器的优先级；
- 指定路由器（DR）和备份指定路由器（BDR）；
- 标识可选性能的5个标记位；
- 始发路由器的所有有效邻居的路由器ID。这个列表仅仅包含这样一些所谓有效的邻居路由器一即在最近的路由器无效时间间隔内，始发路由器接口可以从其接收到Hello数据包的那些邻居。

本小节概述了上面列出的多数信息的含义和用法。随后的章节将会详细

地讲述指定路由器（DR）、备份指定路由器（BDR）、路由器的优先级和阐明Hello数据包的详细格式。当一台路由器从它的邻居路由器收到一个Hello数据包时，它将检验该Hello数据包携带的区域ID、认证信息、网络掩码、Hello间隔时间、路由器无效时间间隔以及可选项的数值是否和接收接口上配置的对应值相匹配。如果它们不匹配，那么该数据包将被丢弃，而且邻接关系也无法建立。

如果所有的参数都匹配，那么这个Hello数据包就被认为是有效的。而且，如果始发路由器的路由器ID已经在接收该Hello数据包的接口的邻居表中列出，那么路由器无效时间间隔计时器将被重置。如果始发路由器的路由器ID没有在邻居表中列出，那么就把这个路由器ID加入到它的邻居表中。

无论何时，路由器发送一个Hello数据包时，都会在这个数据包中列出传送该数据包的链路上所出现的所有邻居的路由器ID。如果一台路由器收到了一个有效的Hello数据包，并在这个Hello数据包中发现了自己的路由器ID，那么这台路由器就认为是双向通信（two-way communication）建立成功了。

一旦双向通信成功建立，邻接关系也就可能建立了。然而，正如前面所提及的，并不是所有的邻居路由器都会成为邻接对象。一个邻接关系的形成与否是依赖于和这两台互为邻居的路由器所连网络的类型的。另外，网络类型也影响OSPF数据包传送的方式。因此，在讲述邻接关系之前，讲述网络类型是必要的。

2. 网络类型

OSPF协议定义了以下5种网络的类型：

- （1）点到点网络（point-to-point）；
- （2）广播型网络（broadcast）；
- （3）非广播多路访问（NBMA）网络；
- （4）点到多点网络（point-to-multipoint）；
- （5）虚链路（virtual links）。

- 点到点网络（point-to-point）

点到点网络，像T1、DS-3或SONET链路，是连接单独一对路由器的。在点到点网络上的有效邻居总是可以形成邻接关系。在这些网络上的OSPF数据包的目的地址也总是保留的D类地址224.0.0.5，这个组播地址称为AllSPFRouters。[\[5\]](#)

- 广播型网络（broadcast）

广播型网络，像以太网、令牌环网和FDDI，也可以更确切地定义为广播型多址网络，以便区别于NBMA网络。广播型网络是多址的网络，因而它们可以连接多于两台设备。而且由于它们是广播型的，所以连接在这种网络上的所有设备都可以接收到个别传送的数据包。在广播型网络上的OSPF路由器正如下面“指定路由器和备份指定路由器”中所讲述的，会选举一台指定路由器和一台备份指定路由器。Hello数据包像所有始发于DR和BDR的OSPF数据包一样，是以组播方式发送到AllSPFRouters（目的地址是224.0.0.5）的。携带这些数据包的数据帧的目的介质访问控制（MAC）地址是0100.5E00.0005。其他所有的路由器都将以组播方式发送链路状态更新数据包和链路状态确认数据包（将在后面讲述）到保留的D类地址224.0.0.6，这个组播地址称为AllDRouters。携带这些数据包的数据帧的目的MAC地址是0100.5E00.0006。

- 非广播多路访问（NBMA）网络

NBMA网络，像X.25、帧中继和ATM等，可以连接两台以上的路由器，但是它们没有广播数据包的能力。一台在NBMA网络上的路由器发送的数据包将不能被其他与之相连的路由器收到。结果是，在这些网络上的路由器有必要增加另外的配置来获得它们的邻居。在NBMA网络上的OSPF路由器需要选举DR和BDR，并且所有的OSPF数据包都是单播的。

- 点到多点网络（point-to-multipoint）

点到多点网络是NBMA网络的一个特殊配置，可以被看作是一群点到点链路的集合。在这些网络上的OSPF路由器不需要选举DR和BDR，OSPF数据包以单播方式发送给每一个已知的邻居。

- 虚链路（virtual links）

虚链路将在后面部分讲述，它可以被路由器认为是没有编号的点到点网络的一种特殊配置。在虚链路上OSPF数据包是以单播方式发送的。

除了以上那5种网络类型外，应该注意的是，所有的网络也都可以归纳到下面两种更普通的网络类型之一：

- 传送网络（**Transit Network**）——与两台或两台以上的路由器相连。这种网络仅仅传送那些“只需仅仅通过”的数据包，也就是这样的一些数据包——它们的始发网络和目的网络都不同于当前的传送网络。
- 末梢网络（**Stub Network**）^[6]——仅仅和一台路由器相连。末梢网络上的数据包总是有一个源地址或者目的地址属于这个末梢网络。也就是说，末梢网络上的所有数据包要么始发于这个末梢网络上的某个设备，要么终止于这个末梢网络上的某个设备。OSPF协议在末梢网络上通告主机路由（就是网络掩码为255.255.255.255的路由）。loopback接口也可以认为是末梢网络，并当作主机路由来通告。^[7]

3. 指定路由器和备份指定路由器

对于OSPF协议来说，在多址网络上有关LSA的泛洪扩散（flooding，将在后面的章节讲述）方面还存在两个问题：

- 在构建相关路由器之间的邻接关系时，会创建很多不必要的LSA。假设在一个多址网络上有 n 台路由器，那么就会构成 $n(n-1)/2$ 个邻接关系，如图8-1。每台路由器都会通告出 $n-1$ 条LSA信息到与之存在邻接关系的邻居路由器，再加上1个网络LSA，这样计算的最终结果是，这个网络上将产生出 n^2 个LSA通告。
- 多址网络本身的泛洪扩散显得比较混乱。某一台路由器向与它存在邻接关系的所有邻居发出LSA，同样地，这些邻接的邻居路由器又向与它有邻接关系的邻居的邻居发出这个LSA，这样将会在一个网络上创建很多个相同LSA的副本。

为了在一个多路访问网络避免这些问题的发生，可以在多路访问网络上选举一台指定路由器。这台指定路由器将完成以下工作：

- 描述这个多路访问网络和OSPF区域内其余与其相连的路由器；
- 管理这个多路访问网络上的泛洪扩散过程。

DR背后的一种概念是广播链路本身被认为是一个“伪节点”，或者虚拟路由器。当SPF树进行计算的时候，把链路看作一个节点，与该链路相连的路由器也是连接到这个节点上的。从与伪节点相连的路由器到这个伪节点的代价就是该路由器与这个广播链路相连的接口的出站代价，但是从伪节点到任何与之相连的路由器的代价都为0。通过这种方式，所有路径的代价都不会受到伪节点的影响。

网络中的每一台路由器都会与DR形成一个邻接关系，如图8-2所示，DR在特定的网络LSA中表示为一个伪节点。请记住，一台路由器可能是它所连接的其中一个多路访问网络的指定路由器，也可能不是它所连接的另一个多路访问网络的指定路由器。换句话说，指定路由器是路由器接口的特性，而不是整个路由器的特性。

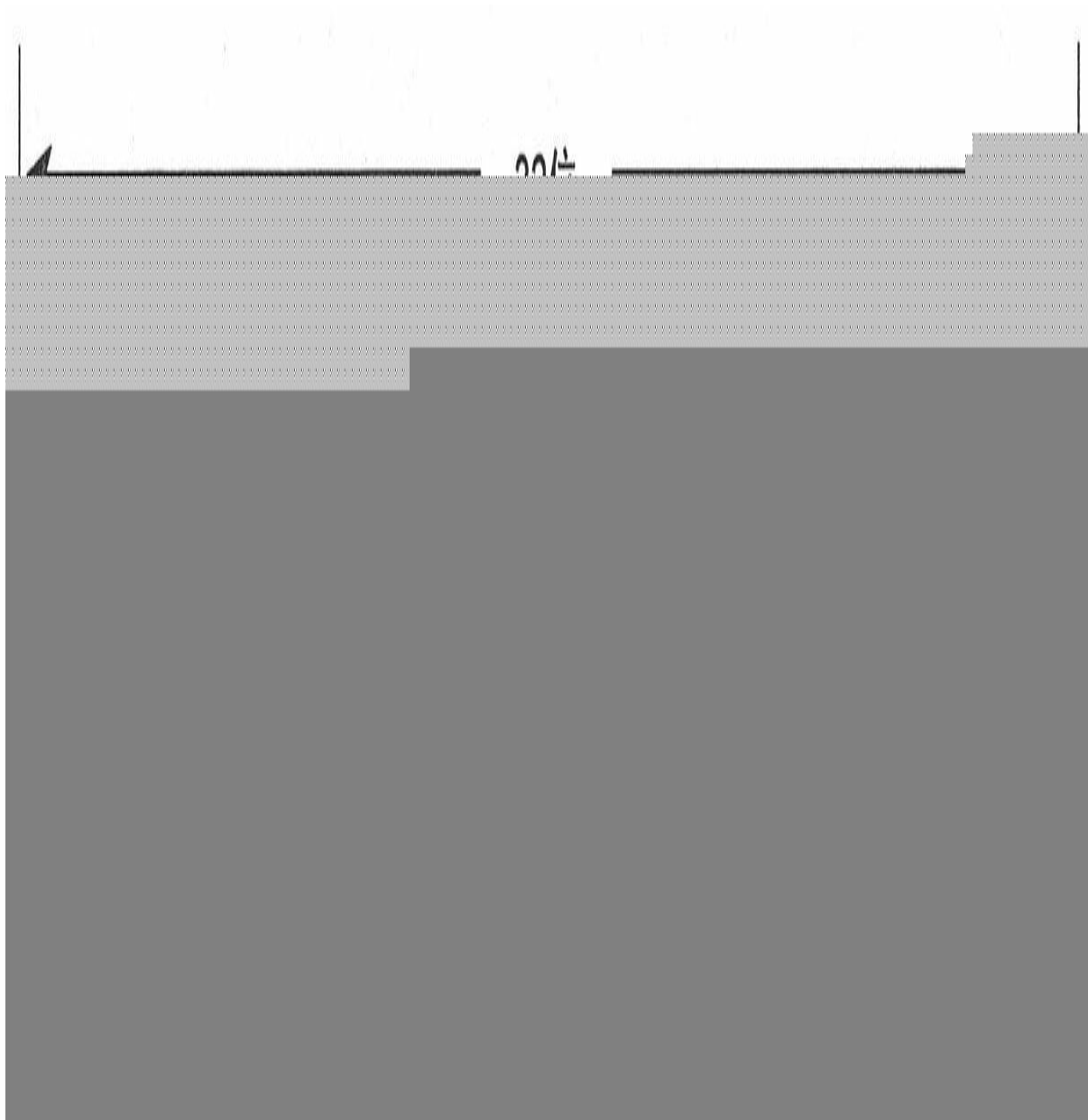


图8-1 在**OSPF**网络上，如果要在每一台路由器和它的邻居路由器之间形成完全网状的**OSPF**邻接关系，那么这5台路由器之间将需要形成**10**个邻接关系；这个网络还将会产生**25**条**LSA**通告

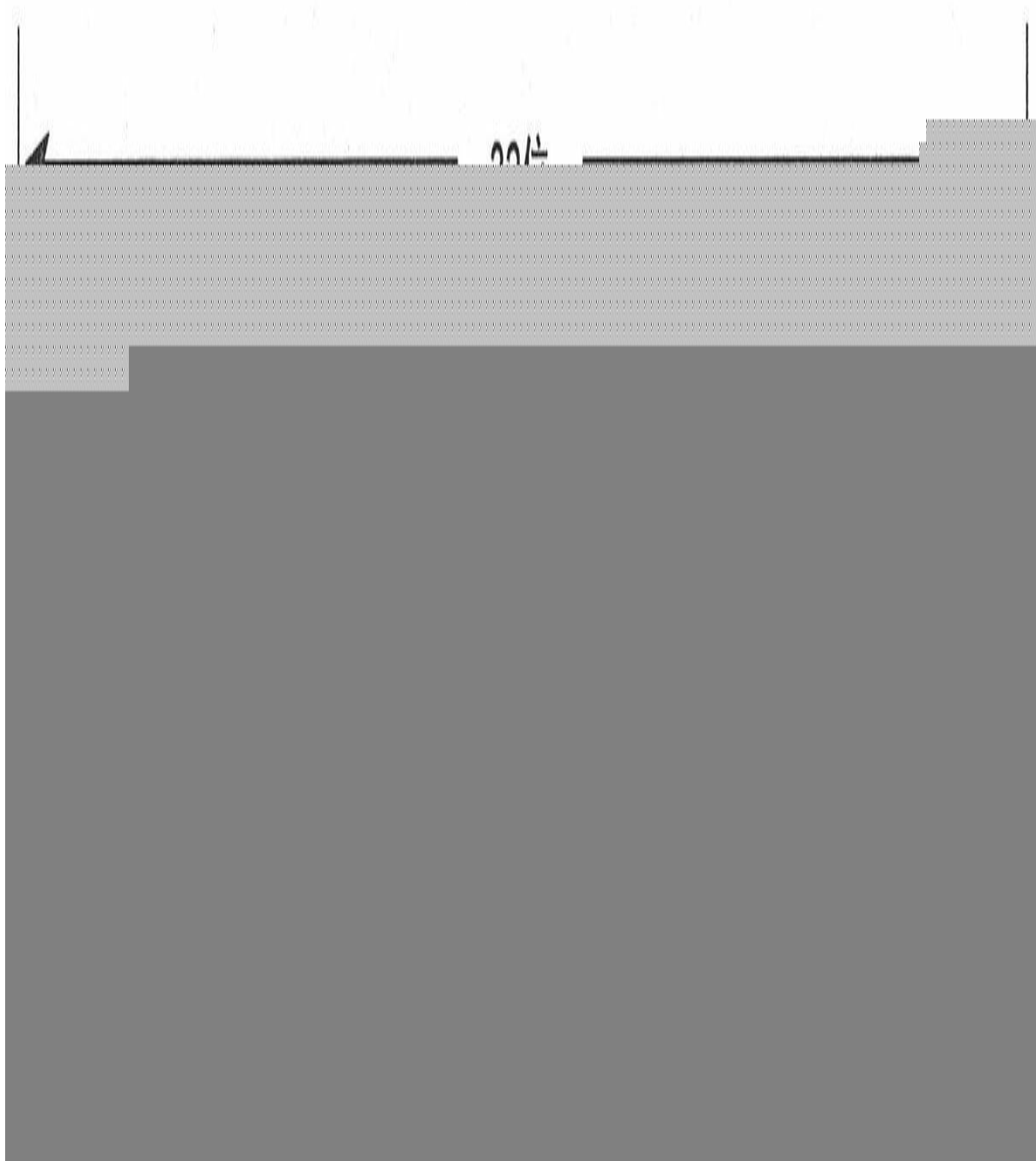


图8-2 指定路由器描述了一个多址网络。网络上的其他路由器都将和这个指定路由器（**DR**）构成邻接关系，而不是它们互相之间构成邻接关系

到目前所描述的为止，可以看出关于指定路由器的一个重要问题是，如果一台指定路由器失效了，就必须选取一台新的指定路由器。同时，网

络上的所有路由器也要重新建立新的邻接关系，并且网络上所有的路由器必须根据新选出的指定路由器进行同步它们的网络数据库（邻接关系创建过程中的一部分）。当所有上述的过程发生时，网络将无法有效地传送数据包。

为了避免这个问题，在网络上除了选取指定路由器，还再选取一台备份指定路由器（BDR）。这样，网络上所有的路由器将和指定路由器（DR）与备份指定路由器（BDR）同时形成邻接关系。DR和BDR之间也将互相形成邻接关系。这时，如果DR失效了，BDR将成为新的DR。但是由于网络上其余的路由器已经和BDR形成了邻接关系，因此网络可以将无法传送数据的影响降低到最小。

DR和BDR的选取是通过一个接口状态机的方式触发的，接口状态机将在后面讲述。为了能够使选取的处理过程可以进行，需要满足以下一些前提条件：

- 每台路由器的每一个多点访问接口都有一个路由器的优先级（Router Priority），用一个8位的无符号整数来表示，范围大小是0~255。在Cisco路由器上，缺省的优先级是1。基于每一个多点访问的接口都可以通过命令**ip ospf priority**来更改。具有0优先级的路由器将不能成为DR或者BDR。
- Hello数据包包含了表示始发路由器指定的路由器优先级的字段，也包含了表示路由器认为可能是DR和BDR的相关接口的IP地址的字段。
- 当一个接口在一个多址网络上开始有效时，它将把它的DR和BDR的地址设置为0.0.0.0。同时它也将等待计时器（wait timer）的值设置为等于路由器无效时间间隔（RouterDeadInterval）。
- 在多址网络上已经存在的接口将把DR和BDR的地址记录入一个接口数据结构表中，接口数据结构表将在后面的章节中讲述。

DR和BDR的选取过程如下描述：

（1）在路由器和它的邻居路由器之间首先建立双向通信（2-way communication），接着检查每台邻居路由器发送的Hello数据包的优先级、DR和BDR字段。列出所有具有DR和BDR选取资格的路由器的列表

（也就是说，路由器的优先级要大于0，并且它的邻居状态至少要是2-way的）；接着，所有的路由器将宣称自己是DR路由器（Hello数据包的DR字段是它们自身接口的地址）；所有的路由器也将宣称它们自己是BDR路由器（Hello数据包的BDR字段是它们自身接口的地址）。除非没有选取资格，路由器计算时也将在这个具有选取资格路由器的列表中包括它本身。

（2）从具有选取资格的路由器的列表中，创建一个还没有宣告为DR路由器的所有路由器的子集（宣告自己为DR路由器的路由器不能被选取为BDR路由器）。

（3）如果在这个子集中的一台或者多台邻居路由器，它们在Hello数据包的BDR字段包含了它们自己的接口地址，那么具有最高优先级的邻居路由器将被宣告为BDR路由器。在优先级相同的条件下，具有最高路由器ID的邻居路由器将被选作BDR路由器。

（4）如果在这个子集中没有路由器宣称自己是BDR路由器，那么具有最高优先级的邻居路由器将被宣告为BDR路由器。在优先级相同的条件下，具有最高路由器ID的邻居路由器将被选作BDR路由器。

（5）如果一台或多台具有选取资格的路由器在Hello数据包的DR字段包含它们自己的接口地址，那么具有最高优先级的邻居路由器将被宣告为DR路由器。在优先级相同的条件下，具有最高路由器ID的邻居路由器将被选作DR路由器。

（6）如果没有路由器宣称自己是DR路由器，那么新选取的BDR路由器将成为DR路由器。

（7）如果正在执行计算的路由器是新选取的DR或BDR路由器，或者它不再是DR或BDR路由器了，那么将重复以上的2~6步骤。

简单地说，当一台OSPF路由器有效（active）并去发现它的邻居路由器时，它将去检查有效的DR和BDR路由器。如果DR和BDR路由器存在的话，这台路由器将接受已经存在的DR和BDR路由器。如果BDR路由器不存在，将执行一个选取过程，选出具有最高优先级的路由器作为BDR路由器。如果存在多台路由器具有相同的优先级，那么在数值上具有最高路由器ID的路由器将被选中。如果没有有效的DR路由器存在，那么BDR路由器将被选举为DR路由器，然后再执行一个选取过程选取BDR

路由器。

这里需要注意的是，路由器的优先级可以影响一个选取过程，但是它不能强制更换已经有效的DR或BDR路由器。也就是说，在已经选取了DR和BDR路由器后，如果一台具有更高优先级的路由器变为有效的，那么这台新的路由器将不会替换DR或BDR路由器的任何一台。因此，在一个多访问网络上，最先初始化启动的两台具有DR选取资格的路由器将成为DR和BDR路由器。

一旦DR和BDR路由器选取成功，其他的路由器（称为DRothers）将只和DR及BDR路由器之间形成邻接关系。所有的路由器将继续以组播方式发送Hello数据包到AllSPFRouters（组播地址是224.0.0.5），因此它们能够跟踪它们的邻居路由器，但是DRothers路由器只以组播方式发送更新数据包到AllDRouters（组播地址是224.0.0.6）。只有DR和BDR路由器去侦听这个地址，反过来，DR路由器将使用组播地址224.0.0.5泛洪扩散更新数据包到DRothers。

请注意，如果在一个多访问网络上只有惟一的一台具有选取资格的路由器相连，那么这台路由器将成为DR路由器，而且在这个网络上没有BDR路由器。其他所有的路由器都将只和这台DR路由器建立邻接关系。如果没有具有选取资格的路由器和一个多访问网络相连，那么这个网络上将没有DR或者BDR路由器，而且也不建立任何邻接关系。在这种情况下，网络上所有路由器的邻居状态都将停留在“2-Way”状态（将在后面的“邻居状态机”部分中阐述）。

关于DR和BDR路由器所需要完成的更多功能将在随后的章节作更多地完整描述。

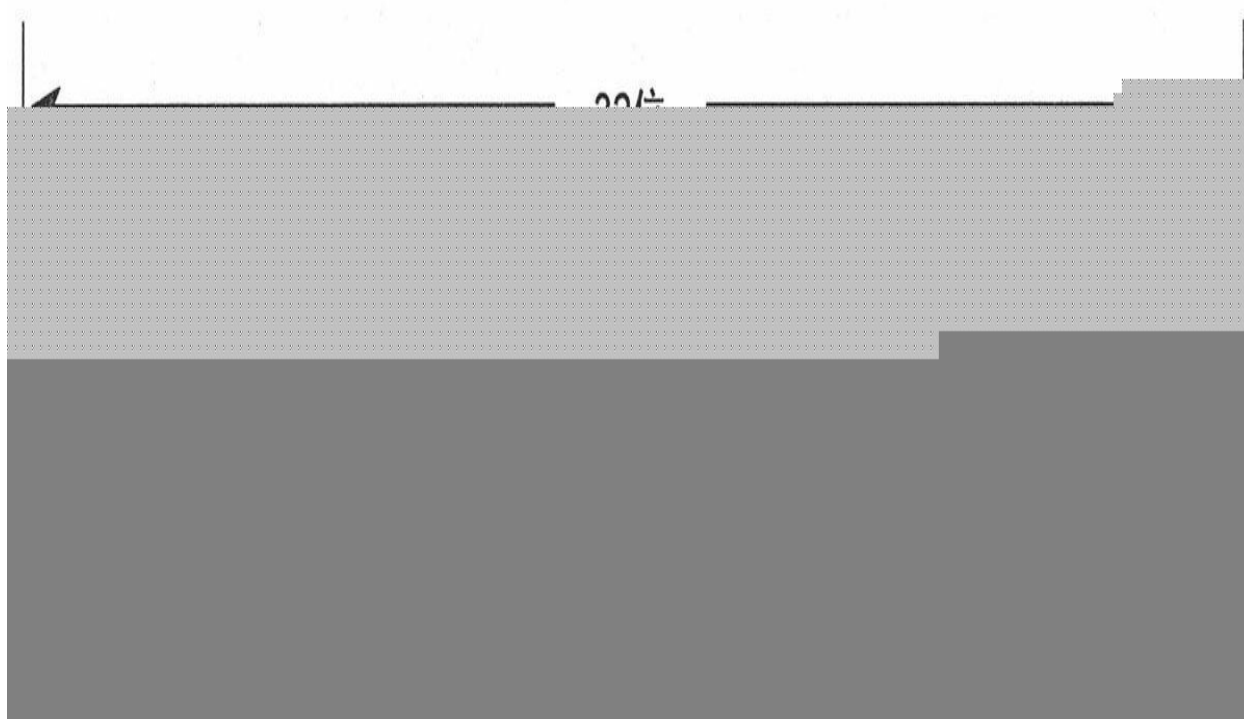
4. OSPF接口

链路状态协议的基本要点是它涉及到了路由器之间的链路和那些链路的状态。在Hello数据包发送及在邻接关系建立之前，以及在LSA通告发送之前，一台OSPF路由器必须了解它自己的链路情况。OSPF协议通过路由器的接口信息来了解链路信息，所以在讲述OSPF协议时会混用接口和链路这两个术语，而不会仔细区分它们的含义。本小节将讲述OSPF协议接口的数据结构和OSPF协议接口的不同状态。

（1）OSPF接口数据结构

运行OSPF协议的路由器将为每一个启动OSPF协议的接口维护一个数据结构。参见示例8-2所示，使用show ip ospf interface命令观察路由器接口的数据结构的内容。[\[8\]](#)

示例8-2 可以使用命令**show ip ospf interface**来观察与路由器接口相关的**OSPF**协议的具体信息。在这个例子中，接口连接到点到点类型的网络



路由器接口的数据结构信息说明如下：

- **IP Address and Mask**（**IP地址和掩码**）——这个信息是路由器接口所配置的IP地址和掩码。始发于这个接口的OSPF数据包将把这个地址作为源地址。参见示例8-2所示，这个地址/掩码组合是192.168.21.21/30。
- **Area ID**（**区域ID**）——就是接口所在的区域，也就是这个接口所属的网络指定的区域ID。始发于这个接口的OSPF数据包将使用这个区域ID。参见示例8-2所示，这里所显示的接口区域ID是7。

- **Process ID**（进程ID）——这个特性是Cisco公司特有的属性，不是OSPF协议开放标准的一部分。Cisco路由器依赖这个特性能够在同一台路由器中运行多个OSPF进程，并且使用这个进程ID来区分这些OSPF进程。进程ID的概念仅在所配置的路由器上有效，而在该路由器之外没有意义。参见示例8-2所示，这里的进程ID是1。
- **Router ID**（路由器ID）——参见示例8-2所示，这里的路由器ID是192.168.30.70。
- **Network Type**（网络类型）——和这个接口相连的网络类型：广播型、点到点类型、NBMA、点到多点类型或虚链路等。在示例8-2中，网络的类型是点到点。[\[9\]](#)
- **Cost**（代价）——是指从该接口发送出去的数据包的出站接口代价。链路代价是OSPF协议的度量，并使用16位无符号的整数表示，范围在1~65535之间。Cisco公司使用的缺省代价是 10^8 /BW，表示为一个整数，在这里BW是指在接口上配置的带宽，而 10^8 是Cisco路由器使用的参考带宽。示例8-2中所示的接口配置了一个128kbit/s的带宽（示例中没有显示），所以它的代价是 10^8 /128kbit/s=781。

路由器接口的代价值可以通过命令**ip ospf cost** 来改变，当在一个多厂商产品的网络环境中配置Cisco路由器时，这个命令变得十分重要。例如，其他厂商的路由器在其所有的接口上使用的缺省代价是1（实际上就是把OSPF的代价映射为跳数）。如果网络中所有的路由器没有使用同一种计算代价的方式来指定OSPF的代价，那么OSPF协议的路由选择将出现不正确的、次优的，或其他一些不确定的情形。

使用 10^8 作为接口的参考带宽在现代一些带宽高于100Mbit/s（例如OC-3或吉比特以太网）的网络介质中会产生一个问题。 10^8 /100M=1，这意味着更高带宽的传输介质在OSPF协议中将会计算出一个小于1的分数，这在OSPF协议中是不允许的。因此，任何代价的计算结果如果是小于1的分数，都将四舍五入为1。但是，如果我们的网络是由高带宽链路组成的，这就意味着所有接口的代价都变成了1，从而计算最短路径就变成了基于最小的路由器跳数了。为了修正这个问题，Cisco提供了命令**auto-cost reference-bandwidth**，该命令允许管理者更改缺省的参考带宽。

其他一些接口数据结构的参数信息如下：

- **InfTransDelay** ——这个信息是指LSA从路由器的接口发送后经历的时间，以秒数计算，当LSA从路由器接口发出后将会引起这个参数值不断地增大。参见示例8-2所示，在图中它是以Transmit Delay来显示的，并且在Cisco路由器上缺省值为1s。InfTransDelay可以通过命令**ip ospf transmit-delay** 来改变。
- **State** （状态）——这个接口的功能状态将在后面的“OSPF接口状态机”部分讲述。
- **Router Priority** （路由器优先级）——用来选择DR和BDR的一个8位无符号整数，范围是0~255。在示例8-2中没有显示路由器的优先级是因为这里的网络类型是点到点的，而在这种网络类型中不需要选取DR和BDR。参见示例8-3所示，它显示了同一台路由器上另一个OSPF接口，从图中可以看出，这个接口与一个广播型网络相连接，因此，在这个网络上将会选取DR和BDR。在Cisco路由器上，路由器的优先级缺省为1，并且可以通过命令**ip ospf priority** 来改变。

示例8-3 这里所显示的接口和一个广播型网络相连，并且这台路由器是该广播型网络上的**DR**

• **Designated Router**（指定路由器）——对于和路由器接口相连的网络的指定路由器（DR），路由器将同时记录下指定路由器的Router ID和它与这个共享网络相连的接口地址信息。注意，在示例8-2中并没有显示出指定路由器，因为只有在多址网络类型中才会显示出指定路由器。在示例8-3中，这里的指定路由器是192.168.30.70，它所连接的接口地址是192.168.17.73。在示例8-3中查看一下Router ID、接口地址和接口状态，就会发现路由器Renoir就是DR。

• **Backup Designated Router**（备份指定路由器）——对于和路由器接口相连的网络的BDR，路由器也将同时记录下指定路由器的Router ID和它与这个共享网络相连的接口地址信息。参见示例8-3，这里的BDR是192.168.30.80，而它所连接的接口地址是192.168.17.74。

• **HelloInterval**——是指在接口上传送两个Hello数据包之间的周期性间隔时间，以秒（s）来表示。这个周期时间是在从接口发送

的Hello数据包中通告的。Cisco路由器在广播型网络上的缺省值为10s，而在非广播型网络上的缺省值为30s，并且可以通过命令**ip ospf hello-interval**来改变。在示例8-3中，HelloInterval是通过Hello表示的，并在这里使用了路由器配置的缺省值。

- **RouterDeadInterval** ——是指在宣告邻居路由器无效之前，本地路由器从与一个接口相连的网络上侦听到来自于邻居路由器的一个Hello数据包所经历的时间，以秒（s）来表示。

RouterDeadInterval是在从接口发送的Hello数据包中通告的。在Cisco公司的路由器上，这个时间缺省的是HelloInterval的4倍，并且可以通过命令**ip ospf dead-interval**来改变。在示例8-3中，RouterDeadInterval是通过Dead表示的，并在这里使用了路由器配置的缺省值。

- **Wait Timer**（等待计时器） ——在开始选取DR和BDR之前，路由器等待邻居路由器的Hello数据包通告DR和BDR的时长。等待计数器的时间长度就是RouterDeadInterval的时间。在示例8-2中的等待时间是没有意义的，因为那个接口是和一个点到点的网络相连的，没有DR和BDR的选取问题。

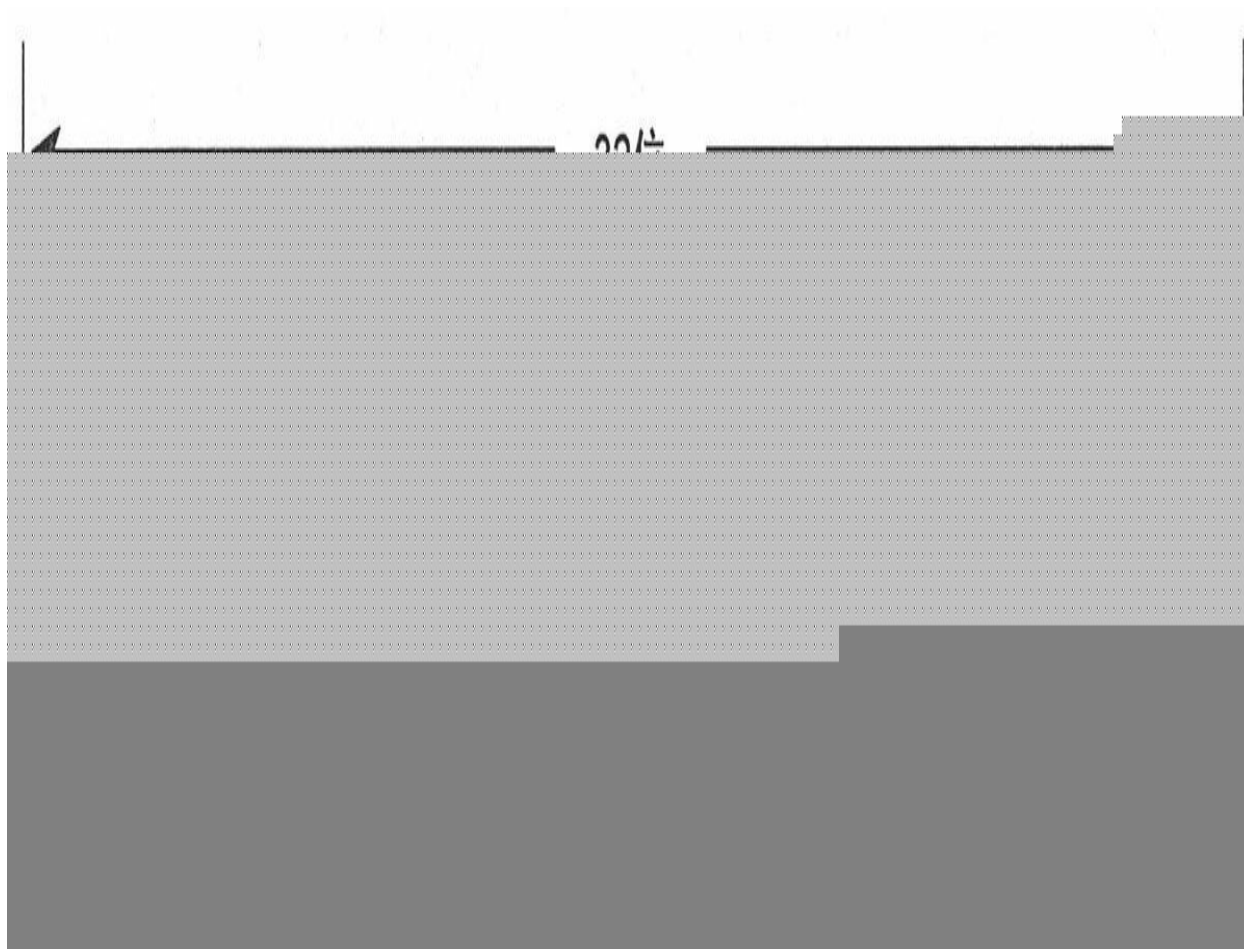
- **RxmtInterval** ——是指在没有得到确认的情况下，路由器重传OSPF数据包将要等待的时间长度，以秒（s）来表示。在示例8-3中，这个时间是通过retransmit来表示的，并在这里使用了Cisco路由器缺省配置的时间——5s。路由器接口的RxmtInterval可以通过命令**ip ospf retransmit-interval**来改变。

- **Hello Timer**（Hello 计时器） ——这个计时器的初始值由HelloInterval来设置。当它计时超时后，路由器将从接口上发送出一个Hello数据包。在示例8-3中显示了Hello Timer将在3s后超时。

- **Neighboring Routers**（邻居路由器） ——是指和这个接口相连的网络上有效邻居路由器（这里的有效邻居路由器是指在RouterDeadInterval的时间内可以收到来自于它们的Hello数据包的那些邻居路由器）的列表。示例8-4显示了同一台路由器另一个接口的信息。在这里，和这个接口相连的网络上应该可以学习到5台邻居路由器，但是只有两台邻居路由器是有邻接关系的（只有建立邻接关系的邻居路由器的Router ID才会显示出来）。作为一个DRother路由器，在这个网络只能和DR与BDR路由器建立邻接关

系，这和多址网络中使用DR的规则是一致的。

示例8-4 在这个网络上，路由器可以看到5台邻居路由器，但是只和DR与BDR路由器建立了邻接关系

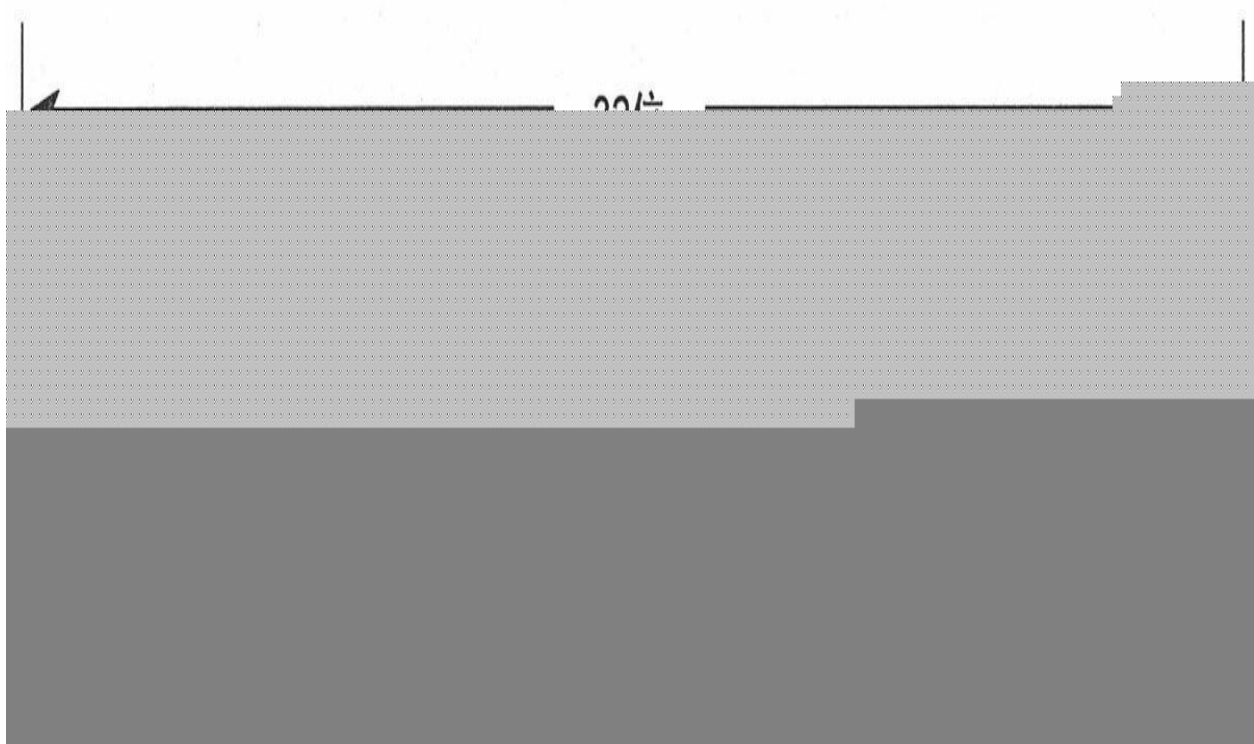


- **AuType** ——描述了在网络上使用的认证类型。OSPF协议认证的类型可以是Null（没有认证）、简单口令或者加密认证（MD认证）。在示例8-4中可以看出这里使用了消息摘要（MD）认证方式。如果使用了Null认证方式，在使用命令**show ip ospf interface**时将不会显示认证方式和密钥信息。
- **Authentication Key**（认证密钥） ——如果在路由器的接口上启用的是简单认证方式，那么认证密钥就是一个64位的口令；如果在路由器的接口上启用的是加密认证方式，那么认证密钥就是一个消息摘要密钥。示例8-4中显示了“youngest key ID”是10。这说明了一个事实，就是加密认证允许在路由器的一个接口上配置多个密钥，

从而可以确保便捷、安全地改变密钥。

示例8-5中显示了一个和NBMA网络相连的接口。注意，在这里HelloInterval的值是NBMA网络类型缺省值30s，而RouterDeadInterval的值缺省是4倍的HelloInterval。

示例8-5 这个接口是和NBMA网络类型的帧中继网络相连的，并且它是这个网络的**BDR**



花费一些时间来比较一下示例8-2～示例8-5中所示的信息会很有收获的。所有这4个接口都是在同一台路由器上，但是它们却又在不同类型的网络环境中，从而扮演不同的角色。在每一个实例中所示的接口状态都表明了不同网络上OSPF路由器的不同角色。在下一个小节中，将会讲述多种接口状态和接口状态机。

（2）OSPF接口状态机

一个启用OSPF协议的接口在它变成完全有效之前，将会在几种接口状态中间发生转换。这些接口状态是失效、点到点、等待、DR、备份、DRother和Loopback等。

- 失效（**Down**）——这是初始化的接口状态。在这个阶段，接口不起任何作用，只是将所有接口的参数设置成它们各自的初始数值，因而，在接口上没有任何路由协议的通信量进行发送和接收。
- 点到点 ——这种接口状态仅仅适用于和点到点、点到多点以及虚电路等网络类型相连的接口。当接口的状态切换到该状态时，这种接口就开始起作用了。这时，路由器的接口将开始每隔HelloInterval的时间发送一次Hello数据包，并尝试和接口链路另一端相连的邻接路由器建立邻接关系。
- 等待（**Waiting**）——这种接口状态仅仅适用于和广播型、NBMA等网络类型相连的接口。当接口的状态切换到这个状态时，这个接口将开始发送和接收Hello数据包，并设置等待计时器的值。而路由器将在接口处于这个状态的时候，试图去识别网络上的DR和BDR。
- 指定路由器（**DR**）——在这种状态下，该路由器是所连网络的指定路由器（DR），并将和所在多址网络上的其他路由器建立邻接关系。
- 备份（**Backup**）——在这种接口状态下，该路由器就是所连网络的备份指定路由器（BDR），并且和所在多址网络上的其他路由器建立邻接关系。
- **DRother** ——在这种接口状态下，该路由器既不是所连网络上的DR路由器，也不是BDR路由器。虽然它将跟踪网络上所有的路由器，但仅仅会和网络上的DR路由器和BDR路由器建立邻接关系。
- **Loopback** ——在这种接口状态下，路由器的接口通过软件或硬件的方式成为环回（loopback）的。虽然一个接口在该种状态下不能发送数据包，但是接口的地址还是可以通过路由器LSA通告出去的（将在后面的部分讲述），因此，进行测试用的数据包能够发现到达这个接口的路径。

图8-3中显示了OSPF接口的状态和引起接口状态发生转换的输入事件。关于输入事件的描述请查看表8-1。

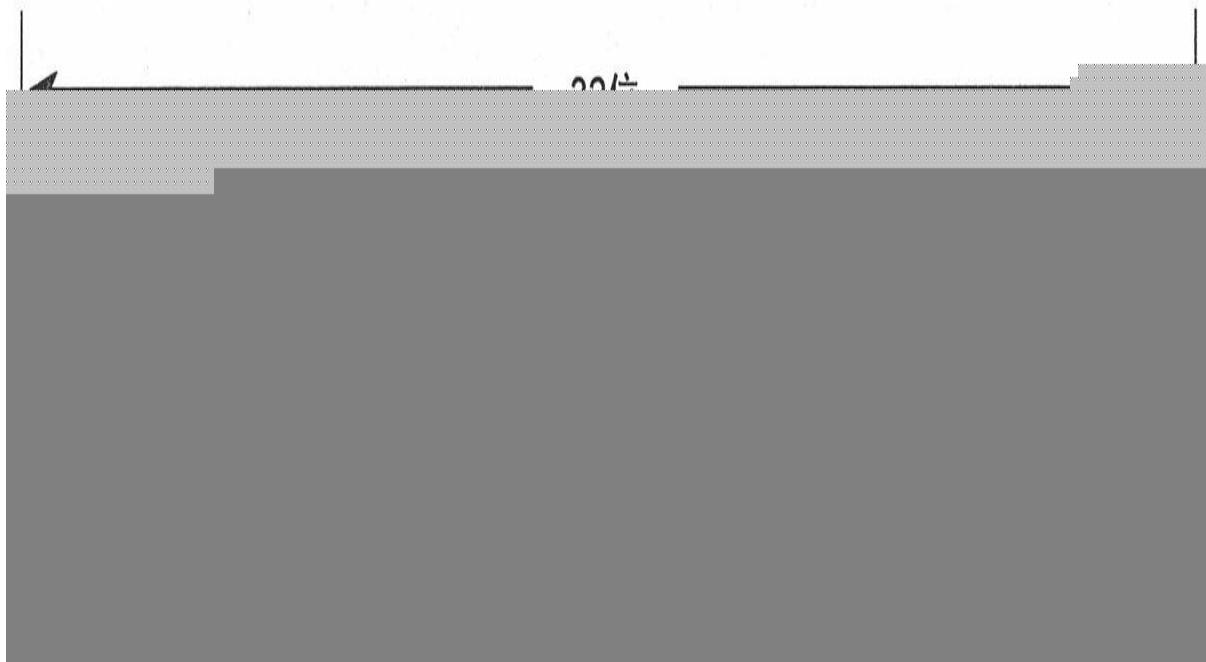


图8-3 OSPF接口状态机；查看表8-1中关于输入事件（IE）的说明

表8-1 接口状态机的输入事件

输入事件	描述
IE1	底层协议指明该网络接口是可操作的
IE2	底层协议指明该网络接口是不可操作的
IE3	网络管理系统或底层协议指明该网络接口打环路后是有效的（looped up）
IE4	网络管理系统或底层协议指明该网络接口打环路后是无效的（looped down）
IE5	收到Hello数据包，在Hello数据包中，要么始发邻居路由器把自身作为BDR列出，要么始发邻居路由器把自身作为DR列出而不指明BDR
IE6	等待计时器超时
IE7	路由器被所在的网络选取为DR路由器
IE8	路由器被所在网络选取为BDR路由器

- IE9 路由器没有被所在网络选取为DR或者BDR路由器
在网络中一组有效的邻居路由器发生了变化。这些变化可能是下列变化之一：
- (1) 和一个邻居路由器之间建立了双向通信
 - (2) 和一个邻居路由器之间丢失了双向通信
 - (3) 收到一个Hello数据包，在该Hello数据包中始发路由器重新把它自身作为DR或BDR路由器列出
- IE10
- (4) 收到来自于DR路由器的Hello数据包，在该Hello数据包中路由器不再把它自身作为DR列出
 - (5) 收到来自于BDR路由器的Hello数据包，在该Hello数据包中路由器不再把它自身作为BDR列出
 - (6) 在RouterDeadInterval超时后，还没有从DR或BDR收到Hello数据包

@@@@@@@@

5. OSPF邻居

前面的章节已经讨论了路由器和与之相连的数据链路之间的关系。虽然一台路由器与其他路由器之间的相互操作和关系在讲述DR和BDR路由器选取的章节中已经做了一些讨论，但是在那些章节介绍DR路由器选取过程的主要目的还是围绕着建立一种与网络的联系。本小节的重点将是讲述网络中的路由器与它的邻接路由器之间的关系。邻居之间建立关联关系的最终目的是为了形成邻居路由器之间的邻接关系，最终可以顺利地传送路由选择信息。

要成功建立一个邻接关系，通常需要下面4个阶段：

- 邻居路由器发现阶段；
- 双向通信阶段（bidirectional communication）——当两台互为邻居的路由器在它们的Hello数据包中都互相列出了它们对方的路由器ID（Router ID）时，路由器就认为双向通信完成了；
- 数据库同步阶段（database synchronization）——路由器之间将进行交换数据库描述（database description）、链路状态请求、链路状态更新和链路状态确认数据包（将在后续的章节讲述）信息，以便确保在邻居路由器的链路状态数据库中包含有相同的数据库信

息。执行这一步骤的目的是使其中一台邻居路由器成为“主路由器”（**master**），而使另一台路由器成为“从路由器”（**slave**）。“主路由器”将控制数据库描述数据包的信息交换；

- 完全邻接阶段（**full adjacency**）。

在前面的介绍中，邻居关系的建立和维持都是通过交换Hello数据包来实现的。在广播类型和点到点类型的网络里，Hello数据包以组播方式发送给组播地址AllSPFRouters（224.0.0.5）。在NBMA类型、点到多点和虚链路类型的网络里，Hello数据包以单播方式发送给每台单独的邻居路由器。单播的发送方式意味着，路由器首先必须知道邻居路由器的存在，这可以通过手工配置的方式或使用逆向地址解析协议（**Inverse ARP**）之类的底层协议来发现。关于在这些网络类型中的邻居的配置方法将会在相应的章节中讲述。

在NBMA类型的网络中，路由器是每经过PollInterval的时间给它邻居状态为down的邻居发送一次Hello数据包，但是在其他的各种网络类型中，路由器都是每经过HelloInterval的时间给它的邻居路由器发送一次Hello数据包。在Cisco路由器中，NBMA网络中PollInterval的缺省值是120s。

（1）邻居数据结构

OSPF路由器在每个OSPF接口的接口数据结构中保存的信息可以用来为每一种类型的网络构成Hello数据包的内容。路由器通过发送包含这些信息的Hello数据包，可以将自己通告给它的邻居路由器。同样的，对于每一台邻居路由器来说，路由器也将维护一个邻居数据结构表，用来表示从其他路由器学习到的Hello数据包的信息。

在示例8-6中，使用命令**show ip ospf neighbor**可以观察到路由器单个邻居的邻居数据结构中的一些信息。[\[10\]](#)

示例8-6 OSPF路由器通过邻居数据结构来描述与每个邻居的每次会话

事实上，每台邻居路由器的数据结构中所记录的信息要比示例8-6中所显示的信息更多。[\[11\]](#)邻居数据结构所含信息如下所述：

- **Neighbor ID**（邻居路由器ID）——邻居路由器的ID。参见示例8-6所示，邻居路由器ID是10.7.0.1。
- **Neighbor IP Address**（邻居IP地址）——是指和网络相连的邻居路由器的接口IP地址。当OSPF数据包以单播方式发送给邻居路由器时，这个地址就是目的地址。参见示例8-6所示，这里的邻居路由器IP地址是10.8.1.2。
- **Area ID**（区域ID）——为了使两台路由器能够互为邻居路由器，路由器收到的Hello数据包中所带的区域ID必须和路由器接收接口配置的区域ID要匹配。示例8-6中所示的邻居路由器的区域ID是0（0.0.0.0）。
- **Interface**（接口）——是指与邻居路由器所在网络相连的接口，也就是说，邻居路由器可以通过该接口到达。在示例8-6中，

该邻居路由器是通过Ethernet0/0接口到达的。

- **Neighbor Priority**（邻居优先级）——这一项表示邻居路由器的优先级，并在邻居路由器的Hello数据包中通告。所谓的优先级是在DR和BDR的选取过程使用的。在示例8-6的邻居路由器的优先级是1，这是Cisco路由器的缺省值。
- **State**（状态）——这一项指的是从本地路由器角度看到的邻居路由器的功能状态，邻居的状态将在下面的“邻居状态机”部分讲述。示例8-6中邻居的状态为Full。
- **Designated Router**（指定路由器）——这个地址包含在邻居路由器发送的Hello数据包的DR字段里面。示例8-6中的DR是10.8.1.1。
- **Backup Designated Router**（备份指定路由器）——这个地址包含在邻居路由器发送的Hello数据包的BDR字段中。示例8-6中的BDR是10.8.1.2。
- **PollInterval**——这个值只用于NBMA网络上相关的邻居路由器。因为在NBMA网络上，邻居路由器可能无法自动地被本地路由器发现，因此，如果邻居状态是失效（Down）的，那么路由器将每经过PollInterval的时间就会发送一个Hello数据包给它的邻居路由器。这里的PollInterval的时长比HelloInterval的时间要长一些。在示例8-6中所示的NBMA网络上的邻居路由器显示出它的PollInterval时间是120s——这是Cisco路由器的缺省值。
- **Neighbor Options**（邻居路由器可选项）——这是邻居路由器支持的一些可选的OSPF性能。关于这些可选项的介绍将在Hello数据包格式的讲述中讨论。示例8-6中可选项字段的值是0x52。
- **Inactivity Timer**（失效计时器）——这是一个时长为RouterDeadInterval（这个参数是在接口数据结构中定义的）的计时器。无论何时，只要从邻居路由器收到一个Hello数据包，这个计时器就会被重新设置。如果在这个失效计时器超时了还没从邻居路由器那里收到一个Hello数据包，那么该邻居路由器将被宣告为失效（down）。参见示例8-6所示，在这里失效计时器用Dead Timer来表示，并且将在30s后超时。

在邻居数据结构中还有一些信息使用命令**show ip ospf neighbor** 没有显示出来，这些没有显示的信息说明如下：

- **Master/Slave**（主/从）——在ExStart状态下，邻居之间协商的主/从关系将用来控制数据库的同步问题。
- **DD Sequence Number**（数据库描述序列号）——是指当前正在向邻居路由器发送的数据库描述序列号。
- **Last Received Database Description Packet**（最后收到的数据库描述数据包）——这个数据包记录了初始化位（Initialize）、后继位（More）和主/从位（Master/Slave）、可选项，以及最后收到的数据库描述数据包的序列号等信息。这个信息可以用来确定下一个数据库描述数据包是否是重复的。
- **Link State Retransmission List**（链路状态重传列表）——这是在邻接关系建立后，OSPF已经进行泛洪扩散（flood）但还没有得到确认的LSA的列表。当LSA还没有得到确认或邻接关系被破坏的时候，LSA将每经过RxmtInterval的时间就重传一次，这里的RxmtInterval是在接口的数据结构中定义的。在示例8-6中显示的不是链路状态重传列表，它显示的是当前列表中LSA的数目（“retransmission queue length”），这里是0。
- **Database Summary List**（数据库摘要列表）——这一项是指在数据库同步期间，数据库描述数据包中向邻居路由器发送的LSA列表。当路由器进入到信息交换状态（Exchange state）时，这些LSA将会构成链路状态数据库。
- **Link State Request List**（链路状态请求列表）——这个列表记录了来自邻居路由器的数据库描述数据包的LSA，这些LSA要比在路由器链路状态数据库中的LSA更加新。而链路状态请求数据包发送给邻居这些LSA的拷贝。当路由器通过链路状态更新数据包收到请求的LSA时，请求列表就会减小，最终将变为空列表。

（2）邻居状态机

OSPF路由器需要邻居路由器在几种邻居状态之间转换后（在邻居数据结构中讲述），才能形成邻居之间的完全邻接关系（Full Adjacent）。

- 失效状态（**Down**）——这是一个邻居会话的初始状态，用来指明在最近一个RouterDeadInterval的时间内还没有收到来自邻居路由器的Hello数据包。除非在NBMA网络中的那些邻居路由器，否则，Hello数据包是不会发送给那些失效的邻居路由器的。在NBMA网络环境中，Hello数据包是每隔PollInterval的时间发送一次的。如果一台邻居路由器从其他更高一些的邻居状态转换到了失效状态，那么路由器将会清空链路状态重传列表、数据库摘要列表和链路状态请求列表。

- 尝试状态（**Attempt**）——这种状态仅仅适用于NBMA网络上的邻居，在NBMA网络上邻居路由器是手工配置的。当NBMA网络上具有DR选取资格的路由器和其邻居路由器相连的接口开始变为有效（Active）时，或者当这台路由器成为DR或BDR时，这台具有DR选取资格的路由器将会把邻居路由器的状态转换到Attempt状态。在Attempt状态下，路由器将使用HelloInterval的时间代替PollInterval的时间来作为向邻居发送Hello数据包的时间间隔。

- 初始状态（**Init**）——这一状态表明在最近的RouterDeadInterval时间里路由器收到了来自邻居路由器的Hello数据包，但是双向通信仍然没有建立。路由器将会在Hello数据包的邻居字段中包含这种状态下或更高状态的所有邻居路由器的路由器ID。

- 双向通信状态（**2-Way**）——这一状态表明本地路由器已经在来自邻居路由器的Hello数据包的邻居字段中看到了它自己的路由器ID，这也就意味着，一个双向通信的会话已经成功建立了。在多址网络中，邻居路由器必须在这个状态或更高状态时才能有资格被选作该网络上的DR或BDR。如果在Init状态下从邻居路由器那里收到一个数据库描述数据包，也可以引起邻居状态直接转换到2-Way状态。

- 信息交换初始状态（**ExStart**）——在这一状态下，本地路由器和它的邻居将建立起主/从关系，并确定数据库描述数据包的序列号，以便为数据库描述数据包的信息交换做准备。在这里，具有最高路由器ID的邻居路由器将成为“主”路由器。

- 信息交换状态（**Exchange**）——在这一状态下，本地路由器将向它的邻居路由器发送可以描述它整个链路状态数据库信息的数据

库描述数据包。同时，在这个Exchange的状态下，本地路由器也会发送链路状态请求数据包给它的邻居路由器，用来请求最新的LSA。

- 信息加载状态（**Loading**）——在这一状态下，本地路由器将会向它的邻居路由器发送链路状态请求数据包，用来请求最新的LSA通告。虽然在Exchange状态下已经发现了这些最新的LSA通告，但是本地路由器还没有收到这些LSA通告。
- 完全邻接状态（**Full**）——在这一状态下，邻居路由器之间将建立起完全邻接关系，这种邻接关系出现在路由器LSA和网络LSA中。

在图8-4～图8-6中，显示了OSPF协议的邻居状态和引起这些邻居状态发生转换的输入事件。在表8-2中对这些输入事件作了详细描述，并在表8-3中定义了点。在图8-4中显示了从最初的功能状态到最完全的功能状态一步一步向更高状态转换的一般过程。在图8-5和图8-6中显示了OSPF协议邻居状态机的整个转换过程。

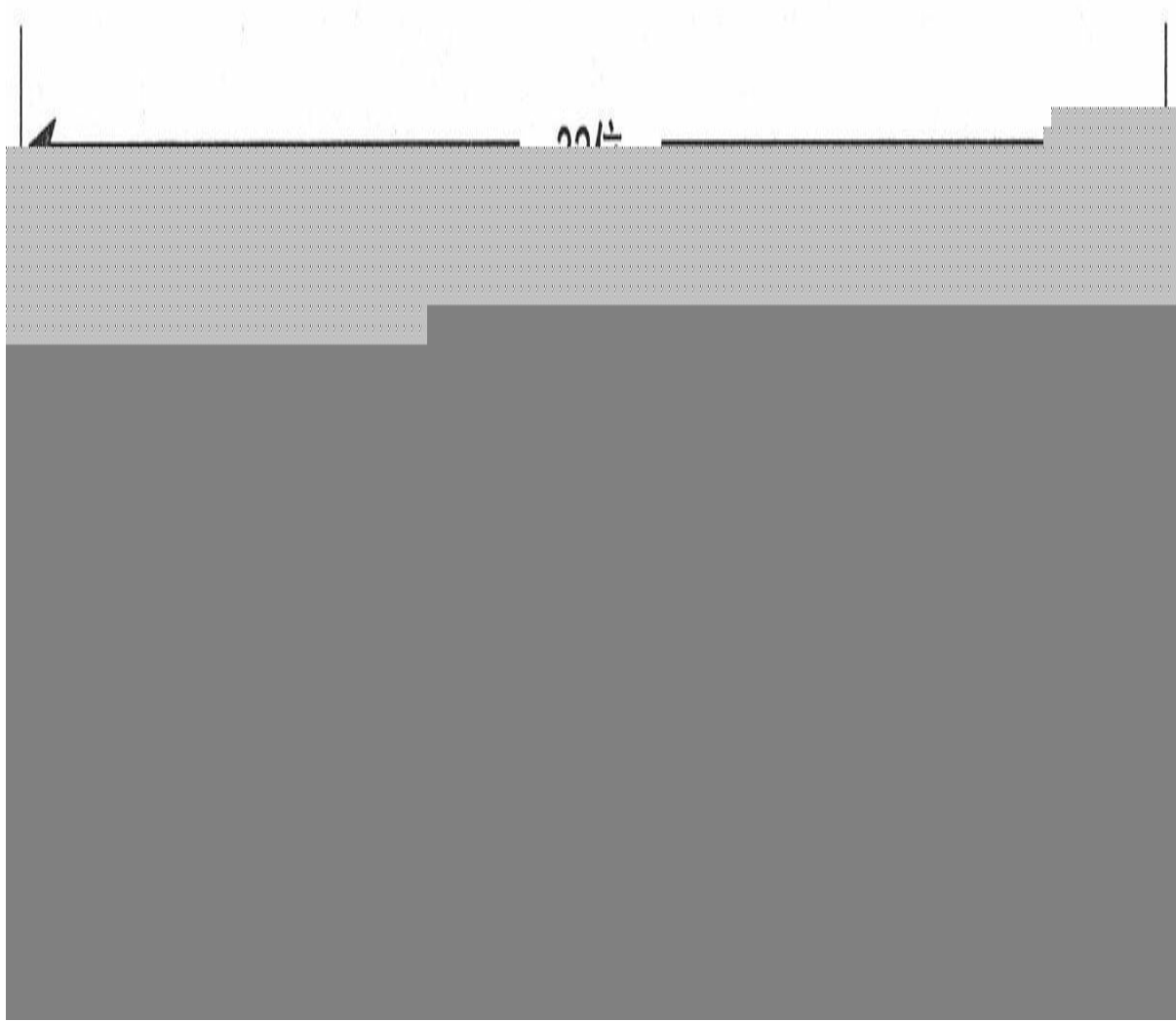


图8-4 在OSPF协议的邻居状态机中，一台邻居路由器从失效状态到完全邻接状态所经过的一系列状态转换

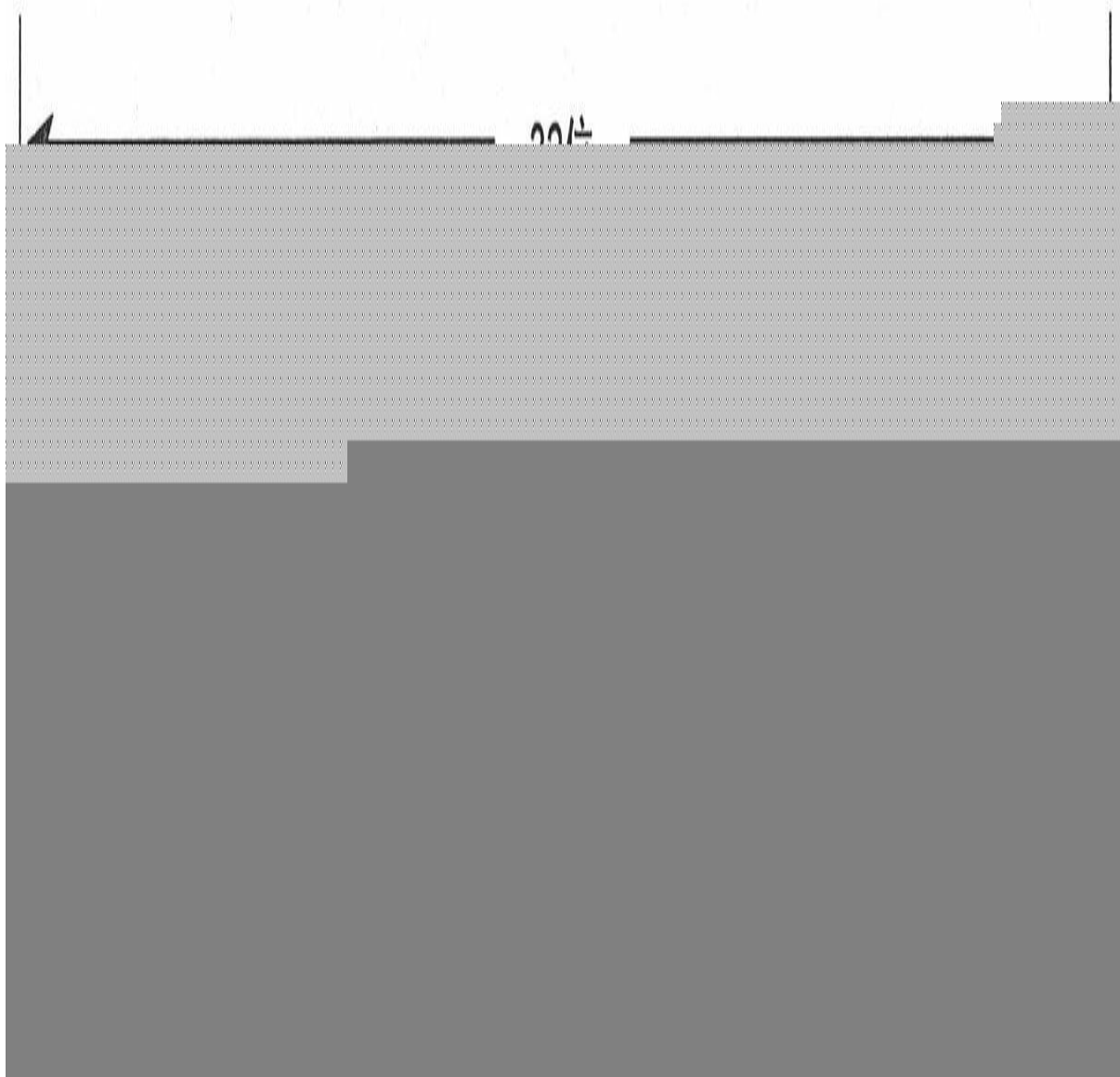


图8-5 在OSPF协议的邻居状态机中，一台邻居路由器从失效状态到初始状态的转换



图8-6 在OSPF协议的邻居状态机中，一台邻居路由器从初始状态到完全邻接状态的转换

表8-2 图8-4、图8-5和图8-6的输入事件

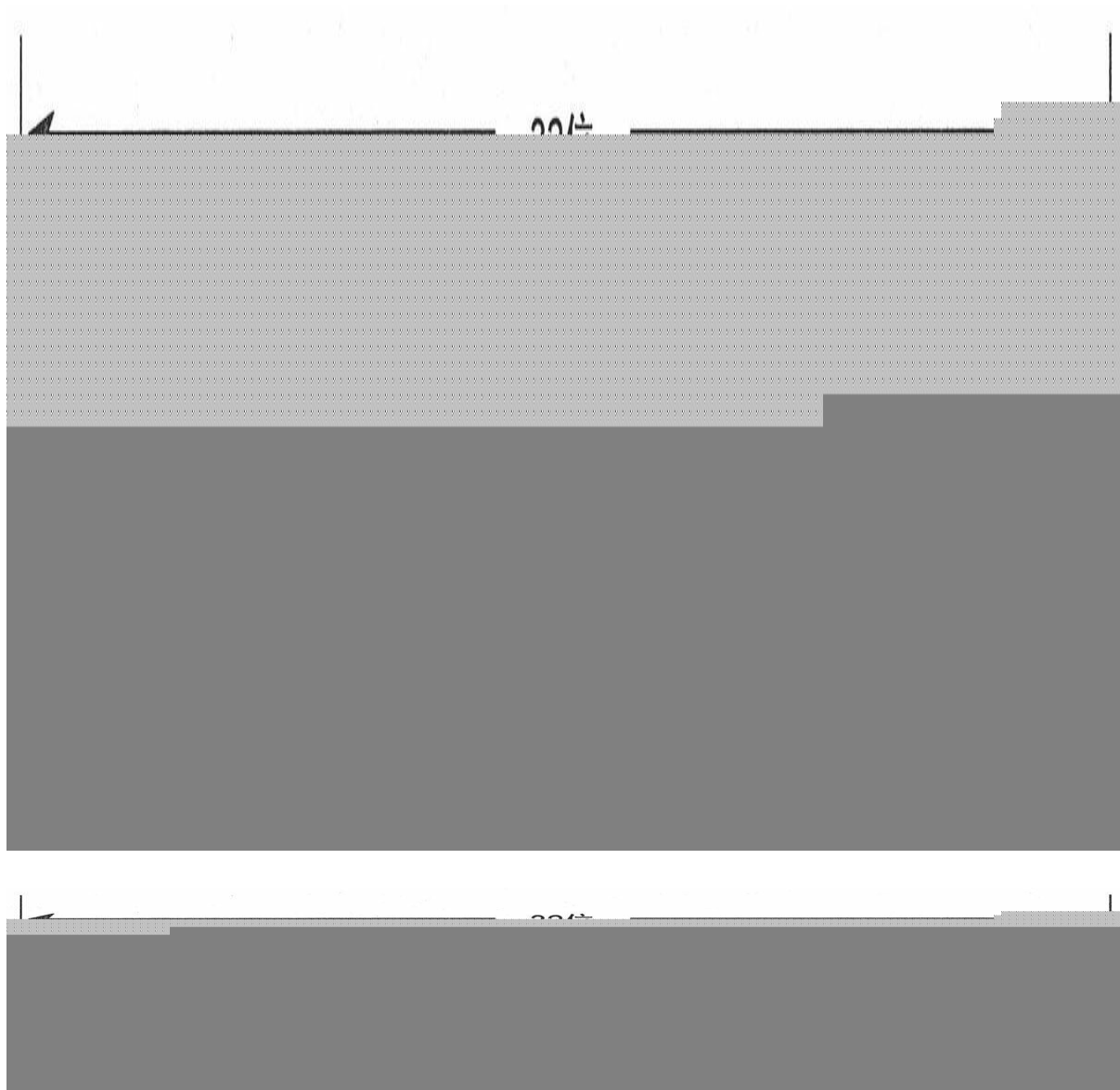


表8-3

图8-4～图8-6中的判定点

(3) 建立一个邻接关系

除非邻居路由器之间Hello数据包的参数不匹配，一般情况下，在点到点、点到多点和虚链路类型的网络上邻居路由器之间总是可以形成邻接关系的。而在广播型网络和NBMA网络上，将需要选取DR和BDR路由器，DR和BDR路由器将和所有的邻接路由器形成邻接关系，但是在DRothers路由器之间没有邻接关系存在。

在一个邻接关系的创建过程中，OSPF协议使用以下3种数据包类型：

- 数据库描述数据包（类型2）；
- 链路状态请求数据包（类型3）；
- 链路状态更新数据包（类型4）。

这些数据包类型的格式将在8.1.7小节中详细讲解。

数据库描述数据包对于邻接关系的建立过程来说非常重要。正如它的名字所暗示的，该数据包携带了始发路由器的链路状态数据库中的每一个LSA的一个简要描述。这些描述不是关于LSA的完整描述，而仅仅是它们的头部——这些信息对于接收路由器判定在它自己的数据库中的LSA通告是否是最新的拷贝来说已经是足够的了。另外，在数据库描述数据包中有3个标记位用来管理邻接关系的建立过程：

- I位，或称为初始位（Initial bit），当需要指明所发送的是第一个数据库描述数据包时，该位设置为1；

- **M位**，或称为后继位（**More bit**），当需要指明所发送的还不是最后一个数据库描述数据包时，该位设置为1；
- **MS位**，或称为主/从位（**Master/Slave bit**），当数据库描述数据包始发于一个“主”路由器时，该位设置为1。

当两台邻居路由器在ExStart状态开始进行主/从关系协商时，它们都将通过发送一个MS位设置为1的空的数据库描述数据包来宣称自己是“主”路由器。这两个数据库描述数据包的数据包描述序列号是由发出这两个数据包的路由器根据当时应该使用到的序列号来确定的。具有较低路由器ID的邻居路由器将成为“从”路由器，并且回复一个MS位设置为0的数据库描述数据包——这个数据库描述数据包的数据包描述序列号设置为“主”路由器的序列号。同时，这个数据库描述数据包也将是第一个携带LSA摘要信息的数据包。当主/从关系协商完成后，邻居状态也将转换到Exchange状态。

在Exchange状态，邻居路由器开始同步它们的链路状态数据库，同步链路状态数据库的操作是通过描述它们各自的链路状态数据库的所有条目来实现的。数据库摘要列表由路由器的链路状态数据库中所有的LSA通告的头部组成，而本地路由器将向它的邻居路由器发送包含这些LSA头部列表的数据库描述数据包。

如果本地路由器发现它的邻居路由器有一条LSA通告不在它自己的链路状态数据库中，或者邻居路由器含有比已知LSA通告更新的拷贝，那么本地路由器将把这条LSA放入它的链路状态请求列表中。随后，本地路由器将发出一个链路状态请求数据包去请求一个关于刚才讨论的LSA的完整拷贝。链路状态更新数据包将会传送这些被请求的LSA的信息。当本地路由器收到关于这些被请求的LSA之后，它将从自己的链路状态请求列表中删除这些LSA的条目。

在更新数据包中传送的所有的LSA必须单独地进行确认。因此，路由器将把这些传送的LSA放入它的链路状态重传列表中。当这些LSA被确认后，路由器就从它的链路状态重传列表中删除它们。LSA可以通过下面两种方法之一来确认：

- **显式确认（Explicit Acknowledgment）**——确认收到包含这个LSA头部的链路状态确认数据包。

- 隐式确认（**Implicit Acknowledgment**）——确认收到包含这个LSA的相同实例（没有其他更加新的LSA）的更新数据包。

“主”路由器将控制数据库的同步过程，并确保每次只有一个数据库描述数据包是未处理的。当“从”路由器收到一个从“主”路由器发出的数据库描述数据包后，“从”路由器将通过发送一个具有相同序列号的数据库描述数据包来确认那个数据包。如果“主”路由器在RxmtInterval（这个参数在接口数据结构一节中已经介绍）的时间内没有收到一个关于未处理的数据库描述数据包的确认，那么“主”路由器将会发送该数据包的一份新拷贝。

“从”路由器发送数据库描述数据包仅仅用来响应它从“主”路由器那里收到的数据库描述数据包。如果它所收到的数据库描述数据包具有一个新的序列号，那么“从”路由器将发送一个具有相同序列号的数据库描述数据包。如果它所收到的数据库描述数据包的序列号和在这之前已确认的数据库描述数据包相同，那么这个确认数据包就是重发的。

当数据库同步过程完成后，将会出现下面两种状态转换的其中一种：

- 如果链路状态请求列表中仍然还有一些LSA条目，那么路由器将把邻居的状态转换到加载（Loading）状态。
- 如果链路状态请求列表为空，那么路由器将会把邻居的状态转换到完全邻接（Full）状态。

如果“主”路由器已经发送过可以完整地描述它自己的链路状态数据库所必要的所有数据库描述数据包，并且从“从”路由器收到一个M位设置为0的数据库描述数据包，那么这时“主”路由器就认为数据库的同步过程已完成。如果“从”路由器接收到一个M位设置为0的数据库描述数据包，并且向“主”路由器发送一个确认的M位也设置为0的数据库描述数据包的话（也就是说，“从”路由器已经完全描述了它自己的链路状态数据库），那么这时“从”路由器就认为数据库的同步过程已完成。由于“从”路由器必须确认每一个收到的数据库描述数据包，因此“从”路由器总是最先得知同步过程完成了。

图8-7中显示了一个邻接关系的创建过程。这个例子是直接从RFC2328中引用而来的。

在图8-7中演示了链路状态数据库同步过程中的下列步骤：

（1）在多路访问的网络上，路由器RT1变为有效状态，并发送一个Hello数据包。由于它还没有学习到任何邻居，因而这个Hello数据包的邻居字段是空的，而DR和BDR字段设置为0.0.0.0。

（2）一旦从路由器RT1收到上面的Hello数据包，路由器RT2就会为RT1创建一个邻居数据结构，并将RT1的状态设置为初始状态（Init）。路由器RT2将发送一个Hello数据包给路由器RT1，并在这个Hello数据包的邻居字段里设置RT1的路由器ID。同样的，作为DR，路由器RT2也将把Hello数据包的DR字段设置成它自己的接口地址。

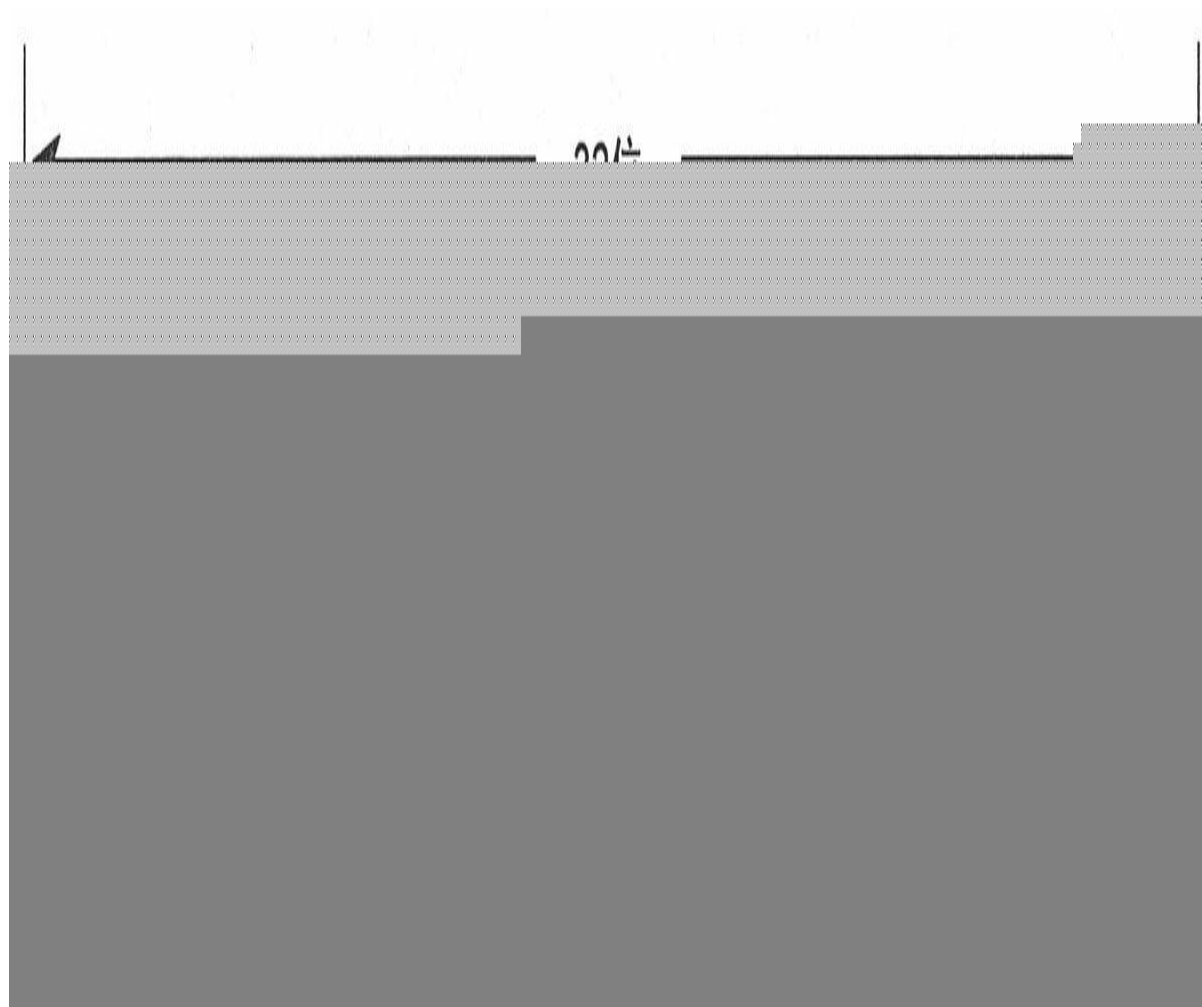


图8-7 链路状态数据库同步过程和相关的邻居状态

（3）当路由器RT1接收到来自路由器RT2的Hello数据包，并看到自己

的路由器ID时（参见表8-2的输入事件IE4），RT1将为路由器RT2创建一个邻居数据结构，并把RT2的状态设置为ExStart状态，以便开始进行主/从关系的协商。接着，路由器RT1产生一个空的数据库描述数据包（没有包含LSA的摘要），并把数据库描述的序列号设置为X。同时设置初始位（I位）来指明这个数据包是路由器RT1用来进行本次信息交换（Exchange）的最初的数据库描述数据包，并设置后继位（M位）来指明这个数据包不是最后的数据库描述数据包，最后还要设置主从位（MS位）来指明路由器RT1声称自己是“主”路由器。

（4）路由器RT2一旦收到来自RT1的数据库描述数据包，就会把RT1的状态转换到ExStart状态。接着，它将发送一个响应的数据库描述数据包，并把这个数据库描述数据包的序列号设置为y。由于路由器RT2拥有比RT1更高的路由器ID，因此它将把自己的MS位设置为1。就像最初的那个数据库描述数据包一样，这个数据包用来进行主从关系协商，因此也是空的。

（5）当这两台邻居路由器同意RT2是主路由器后，路由器RT1就把路由器RT2的状态转换为Exchange状态。路由器RT1将产生一个数据库描述数据包，这个数据包的序列号使用RT2的数据库描述数据包的序列号y，并设置MS位为0用来指明RT1是“从”路由器。同时，该数据包将会传送路由器RT1的链路状态摘要列表中的LSA头部。

（6）路由器RT2一旦收到来自RT1的数据库描述数据包，就会把它的邻居状态转换到Exchange状态。接着，它将发送一个数据库描述数据包，这个数据包包含路由器RT2自己的链路状态摘要列表中的LSA头部，并使它的数据库描述序列号增加到y+1。

（7）当路由器RT1从路由器RT2收到上述的数据库描述数据包后，路由器RT1就会发送一个包含相同序列号的确认数据包。这个过程将一直延续，路由器RT2发送一个单一的数据库描述数据包，接着等待从RT1发出的包含相同序列号的确认数据包，然后RT2再发送下一个数据库描述数据包，直到路由器RT2发出包含最后一个LSA摘要的数据库描述数据包，并把这个数据包的M位设置为0。

（8）收到上述数据包并且确信它所发出的确认数据包包含它自己最后的LSA摘要后，路由器RT1就会认为Exchange过程已经完成。然而，路由器RT1的链路状态请求列表中还存在LSA条目，因此，它将转换到信息加载状态（Loading）。

（9）当路由器RT2收到RT1的最后一个数据库描述数据包时，路由器RT2将把RT1的状态转换为完全邻接状态（Full），这是因为在它的链路状态请求列表中已经没有LSA条目了。

（10）路由器RT1发送链路状态请求数据包，而路由器RT2通过链路状态更新数据包发送被请求的LSA通告，这个过程一直持续到路由器RT1的链路状态请求列表变为空。然后，路由器RT1也将把路由器RT2的状态转换到完全邻接状态。

这里要注意，如果路由器的链路状态请求列表中还有LSA条目，它并不需要等待Loading状态才发送链路状态请求数据包。事实上，当邻居状态还依旧是Exchange状态时路由器就可以发送链路状态请求数据包了。因此，同步过程也许不像图8-7中描绘的那么整齐有序，但是却更有效。

图8-8中显示了使用协议分析仪捕获到的两台邻居路由器之间正在创建邻接关系的过程。虽然链路状态请求数据包和链路状态更新数据包正在被发送，但这时两个邻居仍然处于Exchange状态；注意这里的初始位、后继位、主从位和序列号反映了实际网络环境中的处理过程，这和图8-7中描绘的一般过程是一致的。

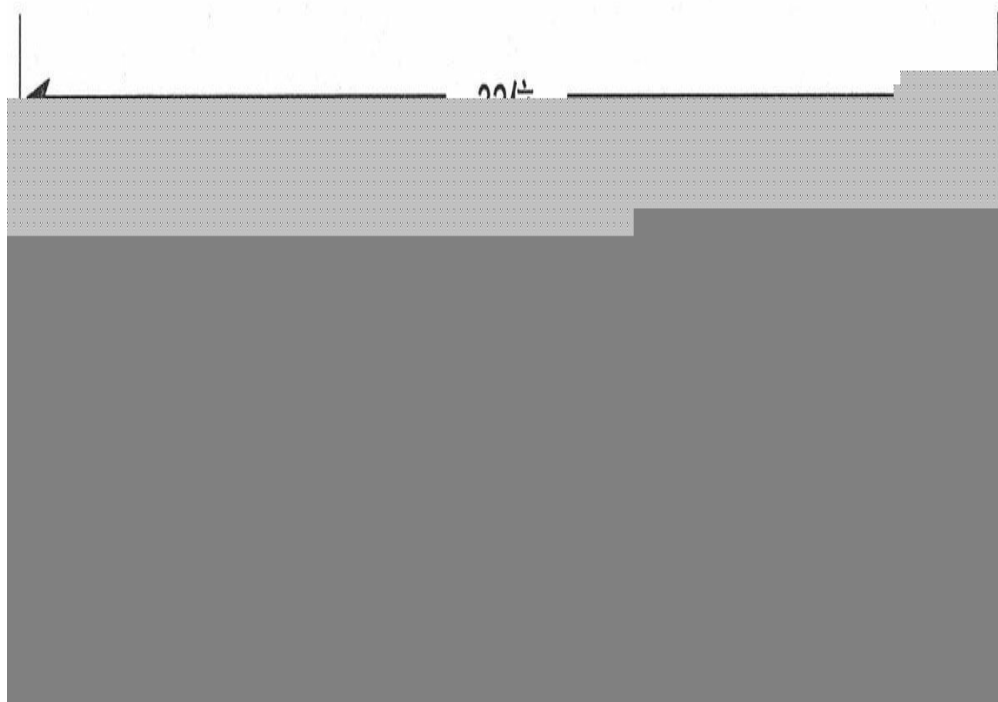
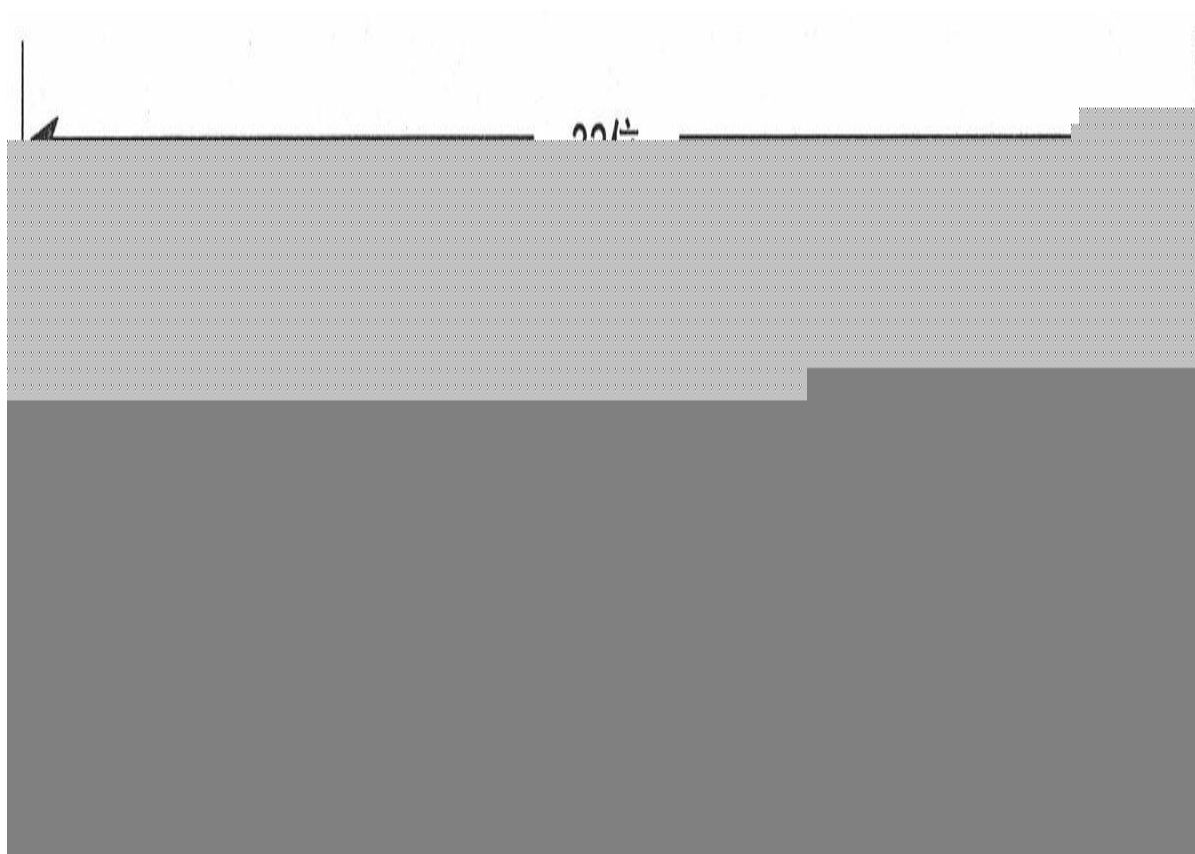


图8-8 这个协议分析仪的捕获界面显示了正在创建邻接关系的过程

在示例8-7中，使用调试命令**debug ip ospf adj**得到的输出结果显示了图8-8中正在创建邻接关系的过程，这是在其中一台路由器（路由器ID是192.168.30.175）上观察到的结果。

示例8-7 调试命令的输出结果显示了图8-8中的邻接事件，这是从其中一台路由器上观察到的结果



在图8-8中，在同步过程结束的时候，可以观察到一系列的链路状态更新数据包和链路状态确认数据包。这些数据包都是LSA泛洪扩散过程的一部分，这将在下面一小节中讲述。

6. 泛洪扩散（**Flooding**）

整个OSPF的拓扑图可以描绘成一组互连的路由器或一组互连的节点，这里所说的互连不是指物理的链路而是指逻辑的邻接关系（如图8-9所示）。为了使这些节点能够在这个逻辑拓扑上完全地进行路由选择，每

一个节点都必须拥有一个关于该拓扑结构的相同拓扑图。这个拓扑图就是拓扑数据库。

OSPF拓扑数据库更熟知的一种叫法是链路状态数据库。这个数据库由路由器可以接收到的所有LSA组成。在拓扑图中发生的一个变化将可以表示为一条或多条LSA的变化。泛洪扩散（Flooding）过程就是将这些变化的或新的LSA发送到整个网络中去，以确保每一个节点的数据库都可以更新，最终保持所有其他节点的数据库的统一性。

泛洪扩散过程将会使用到下面两种类型的OSPF数据包：

- 链路状态更新数据包（Link State Update packets，类型4）；
- 链路状态确认数据包（Link State Acknowledgment packets，类型5）。

正如图8-10中所显示的那样，每一个链路状态更新数据包和确认数据包都可以携带多个LSA。虽然LSA本身是泛洪扩散到整个网络的，但是更新数据包和确认数据包却只在具有邻接关系的两个节点之间传送。

在点到点的网络中，路由器是以组播方式将更新数据包发送到组播地址AllSPFRouters（224.0.0.5）的。在点到多点和虚链路的网络上，路由器是以单播方式将更新数据包发送到邻接邻居的接口地址的。

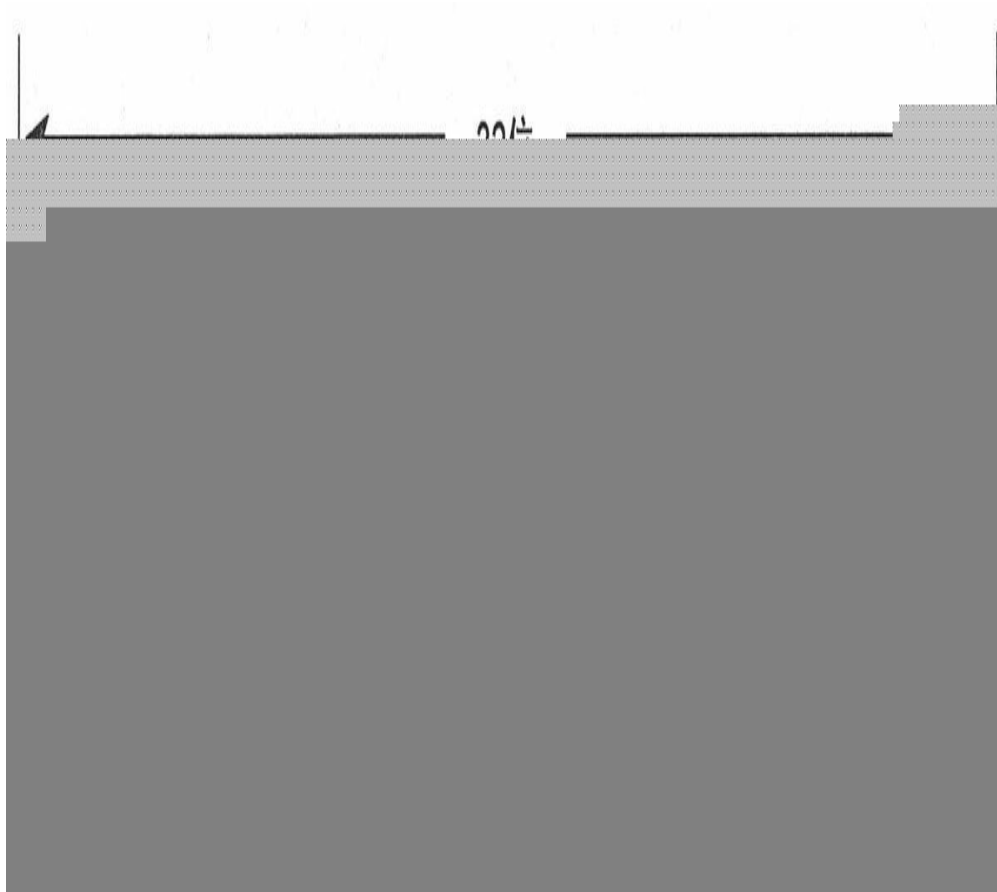


图8-9 **OSPF**协议把一组通过数据链路相连的路由器看作是一组逻辑上通过邻接关系相连的节点



图8-10 **LSA**可以在链路状态更新数据包中发送，从而穿过节点之间的邻接

在广播型的网络上，DRothers路由器只能和DR与BDR路由器形成邻接关系，因此，更新数据包将发送到组播地址AllDRouters（224.0.0.6）。相应地，DR路由器也将以组播方式发送包含LSA的更新数据包到网络

上所有与之建立邻接关系的路由器，这里使用的组播地址是 AllSPFRouters。接着，所有的路由器将从所有其他的接口上泛洪扩散 LSA（如图8-11所示）。虽然BDR路由器也使用组播方式收到和记录了来自DRouters路由器的LSA通告，但是它不会再重复泛洪扩散或者确认这些LSA，除非DR路由器失效了它才会这么做。在NBMA网络上存在同样的DR/BDR的功能特性，只是LSA是以单播方式从DRouters路由器发送给DR和BDR路由器的，并且DR路由器也是以单播方式发送该LSA的拷贝到所有与之建立邻接关系的邻居路由器的。

因为完全相同的链路状态数据库信息是正确操作OSPF最基本的前提，因此LSA的泛洪扩散必须是可靠的。发送LSA的路由器必须要确认它们发出的LSA是否被成功接收了，而接收LSA的路由器也必须确认它们正在接收的LSA信息是正确的。

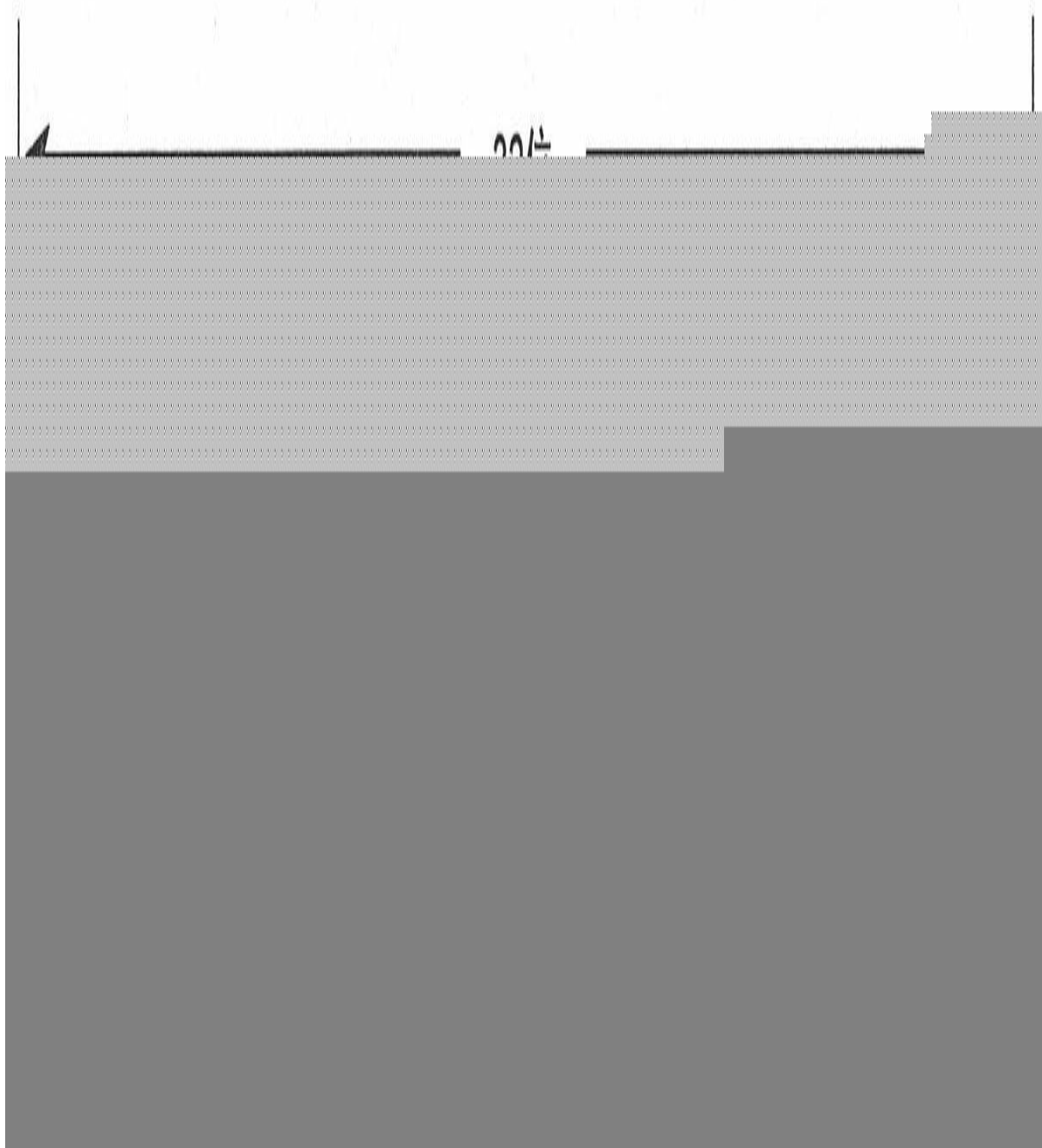


图8-11 在广播型网络上，**DR**路由器只向**DR**和**BDR**路由器发送**LSA**（**a**）；而**DR**路由器将再把这个**LSA**泛洪扩散到所有的与之有邻接关系的邻居路由器（**b**）；接着，所有的路由器在它们其他所有的接口上泛洪扩散这个**LSA**（**c**）

（1）可靠的泛洪扩散：确认

对于可靠的泛洪扩散来讲，每一个单独传送的LSA都必须被确认。在这里，确认可以有隐式确认（Implicit Acknowledgment）或显式确认（Explicit Acknowledgment）两种方式。

邻居路由器可以通过向始发更新数据包的路由器回送包含那个LSA的拷贝信息的更新数据包，来作为对所收到的LSA的隐式确认。在一些情况下，隐式确认方式比显式确认方式更有效。例如，当邻居路由器正打算向始发路由器发送更新数据包的时候。

邻居路由器的显式确认是指通过发送一个链路状态确认数据包来确认收到LSA的方式。而且，可以使用单个链路状态确认来确认多个LSA通告。这个链路状态确认数据包不需要携带完整的LSA信息，而只是需要携带LSA的头部就足以完全识别这些LSA了。

当一台路由器开始发送一个LSA时，会把这个LSA的一份拷贝放进它所发送的每个邻居的链路状态重传列表中。这个LSA通告每隔RxmtInterval的时间重传一次，一直到该LSA得到确认，或者一直到这个邻接关系中断。不论是哪一种网络类型，包含重传的链路状态更新数据包总是以单播方式发送。

确认可以是有时延的（delayed）或直接（direct）的。通过延迟一个确认的方法，更多的LSA通告可以通过单个链路状态确认数据包来确认。在一个广播型的网络上，来自多台邻居路由器的LSA可以由单个组播的链路状态确认数据包来确认。一个被延迟的确认数据包的延迟时间必须小于RxmtInterval的时长，从而避免不必要的数据包重传。一般的情况下，在不同的网络类型上使用于链路状态更新数据包的单播/组播地址约定也可以适用于链路状态确认。

直接的确认总是立即发送并且是以单播方式发送的。直接的确认将在出现下面的两种情况下发送：

- 从邻居路由器收到了重复的LSA，可能表明邻居还没有收到这个LSA的一个确认。
- LSA的老化时间（age）达到最大生存时间（MaxAge，将在下一小节介绍），说明在接收路由器的链路状态数据库里已经没有这个LSA的实例（instance）。

（2）可靠的泛洪扩散：序列号、校验和、老化时间

每一个LSA都包含3个值用来确保在每个数据库中保存的LSA是最新的。这3个数值是序列号、校验和以及老化时间（age）。

OSPF协议使用线性的序列号空间（已在第4章中讲述）和32位有符号的序列号，这里序列号的大小从InitialSequenceNumber（0x80000001）到MaxSequenceNumber（0x7fffffff）。当一台路由器始发一条LSA通告时，它将设置这个LSA的序列号为InitialBequenceNumber。每当这台路由器产生该LSA的一个新实例（Instance）时，该路由器就会将它的序列号增加1。

如果当前LSA的序列号最大值是MaxSequenceNumber并且又必须创建这个LSA的一个新实例时，这台路由器就必须开始从所有的数据库中清除老的LSA。这一操作是通过设置现有LSA的年龄或老化时间（age）为最大生存时间（MaxAge，将在后面的章节介绍），并且重新泛洪扩散它到所有的邻接节点来实现的。一旦所有的邻接邻居路由器确认过这个“提前老化”的LSA后，也就可以泛洪扩散这个LSA的一个含有InitiaBequenceNumber序列号的新实例。

校验和是一个使用Fletcher算法计算得到的16位整数。[\[12\]](#)这个校验和的计算除了Age字段（因为这个age字段在LSA从一个节点到另一个节点时都会发生变化，因此如果校验和也计算这个字段的话，将在每一个节点上都需要重新计算校验和）外，将覆盖整个LSA数据包。驻留在链路状态数据库中的每个LSA的校验和每5min将检验一次，以便确保这个LSA在数据库中没有被破坏。

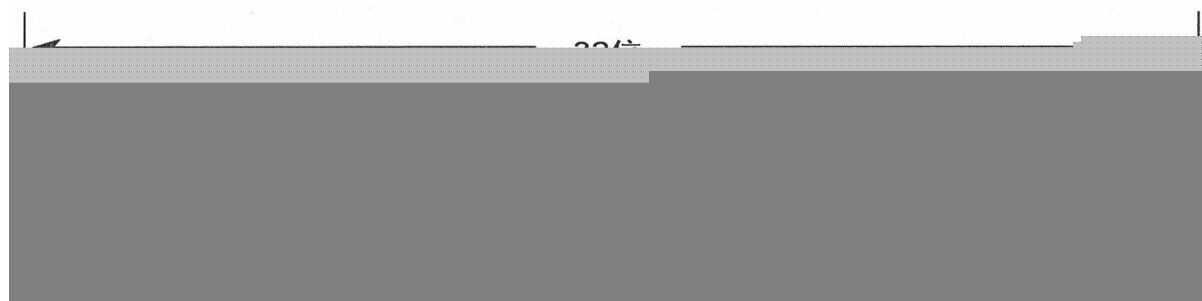
老化时间（age）是一个用来指明LSA的生存时间的16位无符号整数，以秒为单位计，范围是0~3600（1h，也就是最大生存时间）。一台路由器在始发一个LSA时，它就把老化时间设置为0。而当泛洪扩散的LSA经过一台路由器时，LSA的老化时间就会增加一个由InfTransDelay设定的秒数。在Cisco路由器中，InfTransDelay设定的缺省值为1s，这个数值可以通过命令**ip ospf transmit-delay**来改变。当LSA驻留在路由器的数据库中时，LSA的老化时间同样也会增大。

当一条LSA通告的老化时间达到最大生存时间时，LSA将被重新泛洪扩散，并且随后会从路由器的数据库中清除该条LSA。当一台路由器需要从所有路由器的数据库中清除一条LSA时，它会提前把这条LSA的老化

时间设置为最大生存时间并重新泛洪扩散这个LSA。在这里，只有始发这条LSA的路由器才可以提前使这条LSA老化。

示例8-8中显示了一个链路状态数据库的部分信息，从中可以观察到每一条LSA的老化时间、序列号和校验和。有关链路状态数据库和不同类型的LSA的详细讨论将在8.1.3小节中介绍。

示例8-8 在链路状态数据库中记录了每一条LSA的老化时间、序列号和校验和。老化时间是以秒来计算的



当收到某条相同的LSA的多个实例时，路由器将通过下面的算法来确定哪个是最新的LSA实例：

1. 比较LSA实例的序列号。拥有最大的序列号的LSA就是最新的LSA。
2. 如果LSA实例的序列号相同，那么将会比较它们的校验和。拥有最大的无符号校验和的LSA就是最新的LSA。
3. 如果LSA实例的校验和也相同，那么将进一步比较它们的老化时间。如果只有一条LSA拥有大小为最大生存时间的老化时间，那么就认为这条LSA是最新的LSA。
4. 如果这些LSA的老化时间之间的差别多于15min（称做MaxAgeDiff），那么拥有较小的老化时间的LSA将是最新的LSA。
5. 如果上述的条件都无法区分最新的LSA，那么这两个LSA就被认为是相同的。

8.1.2 区域（Area）

到目前为止，读者应该对OSPF协议有一定的了解了。OSPF协议由于使

用了多个数据库和复杂的算法，因而相比前面几章介绍的路由选择协议而言，它将会耗费路由器更多的内存和更多的CPU处理能力。当网络的规模不断增大时，对路由器的性能要求就会显得比较重要甚至达到了路由器性能的极限。另一方面，虽然LSA的泛洪扩散比RIP协议中周期性的、全路由表的更新更加有效率，但是对于一个大型的网络来说，它依然给大量数据链路带来了无法承受的负担。SPF算法本身并没有特别的解决办法。LSA的泛洪扩散和数据库的维护等相关的处理仍然大大加重了CPU的负担。

OSPF协议可以利用区域的概念来缩小这些不利的影响。在OSPF协议的环境下，区域（Area）是一组逻辑上的OSPF路由器和链路，它可以有效地把一个OSPF域分割成几个子域（如图8-12所示）。在一个区域内的路由器将不需要了解它们所在区域外部的拓扑细节。在这种环境下：

- 路由器仅仅需要和它所在区域的其他路由器具有相同的链路状态数据库，而没有必要和整个OSPF域内的所有路由器共享相同的链路状态数据库。因此，在这种情况下，链路状态数据库大小的缩减就降低了对路由器内存的消耗。
- 链路状态数据库的减小也就意味着处理较少的LSA，从而也就降低了对路由器CPU的消耗。
- 由于链路状态数据库只需要在一个区域内进行维护，因此，大量的LSA泛洪扩散也就被限制在一个区域里面了。

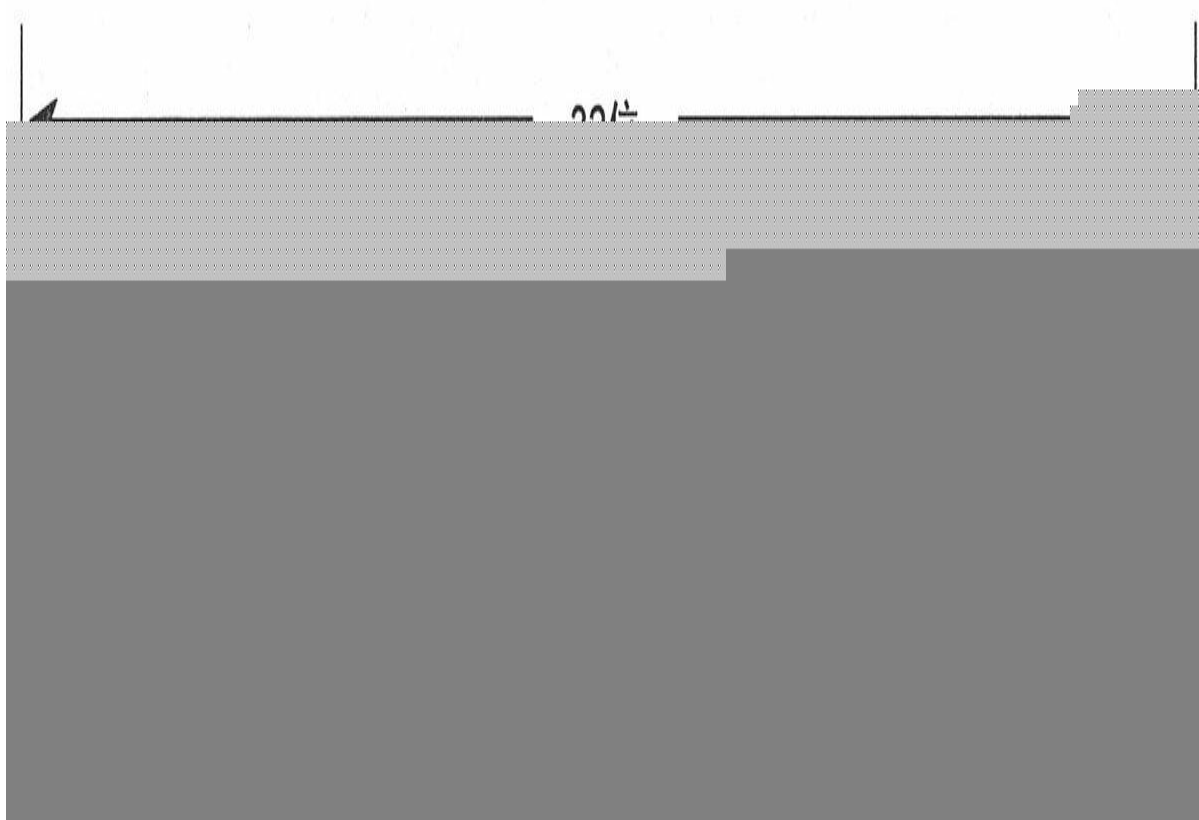


图8-12 OSPF区域是一组逻辑上的OSPF路由器。每一个区域都是通过它自己的链路状态数据库来描述的，而且每台路由器也都只需要维护路由器本身所在的区域的链路状态数据库

区域是通过一个32位的区域ID（Area ID）来识别的。正如图8-12所显示的，区域ID可以表示成一个十进制的数字，也可以表示成一个点分十进制的数字。在Cisco路由器中这两种表示方式都可以使用。到底选用哪一种格式来标识一个具体的区域ID，通常是根据使用的方便性来选择。例如，区域0和区域0.0.0.0的使用效果是相同的，区域16和0.0.0.16，区域271和0.0.1.15，等等。在上述这些实例当中，我们应该首先选用十进制的表示方式。然而，如果要在区域3232243229和区域192.168.30.29两种表示方式中选择一种格式的话，那么后面一种格式可能是比较好的一种选择。

定义了下面3种与区域相关的通信量的类型：

- 域内通信量（**Intra-Area Traffic**）——是指由在单个域内的路由器之间交换的数据包构成的通信量。

- 域间通信量（**Inter-Area Traffic**）——是指由在不同区域的路由器之间交换的数据包构成的通信量。
- 外部通信量（**External Traffic**）——是指由OSPF域内的路由器和其他路由选择域的路由器之间交换的数据包构成的通信量。

区域0（或者区域0.0.0.0）是为骨干域保留的区域ID号。骨干区域（**Backbone Area**）的任务是汇总每一个区域的网络拓扑到其他所有的区域。正是由于这个原因，所有的域间通信量都必须通过骨干区域，非骨干区域之间不能直接交换数据包。

大多数OSPF协议的设计者对于单个区域所能支持的路由器的最大数量都有一个个人认为较适当的粗略的经验值。单个区域所支持的路由器最大数量的范围大约是30~200。但是，在一个区域内实际加入的路由器数量要比单个区域所能容纳的路由器最大数量小一些。这是因为还有更为重要的一些因素影响这个数量，诸如一个区域内链路的数量、网络拓扑的稳定性、路由器的内存和CPU性能、路由汇总的有效使用和注入到这个区域的汇总LSA的数量，等等。正是由于这些因素，有时在一些区域里包含25台路由器可能都已经显得比较多了，而在另一些区域内却可以容纳多于500台的路由器。

只使用单个区域来设计一个小型的OSPF网络是非常合理的。不论区域数量的多少，如果一个区域的数据链路非常之少，以至于没有冗余链路存在的话，那么一些潜在的故障就会发生。如果这样一个区域被分割开来，那么就有可能使网络的通信服务中断。被分割的区域（或称为分段区域，**Partitioned Area**）将在后面的小节中更详细地介绍。

1. 路由器的类型

路由器也像通信量一样可以被分成和区域相关的几个类型。所有的OSPF路由器都是下面4种路由器类型中的一种，如图8-13所示。

- 内部路由器（**Internal Router**）——是指所有接口都属于同一个区域的路由器。
- 区域边界路由器（**Area Border Routers, ABR**）——是指连接一个或者多个区域到骨干区域的路由器，并且这些路由器会作为域间通信量的路由网关。因而，ABR路由器总是至少有一个接口是属

于骨干区域的，而且必须为每一个与之相连的区域维护不同的链路状态数据库。正因为这个原因，ABR路由器通常需要比一般的内部路由器更多的内存和更高性能的路由处理器。ABR路由器将会汇总与它相连的区域的拓扑信息给骨干区域，然后又将这些汇总信息传送给其他的区域。

- 骨干路由器（**Backbone Router**）——是指至少有一个接口是和骨干区域相连的路由器。这个定义意味着ABR路由器也可以是骨干路由器，但是，如图8-13中显示，并不是所有的骨干路由器都是ABR路由器。另外，如果一台内部路由器的所有接口都属于区域0，那么这台内部路由器也是一台骨干路由器。

- 自主系统边界路由器（**Autonomous System Boundary Router, ASBR**）——可以认

为是OSPF域外部的通信量进入OSPF域的网关路由器，也就是说，ASBR路由器是用来把其他路由选择协议（例如，图8-13中显示的BGP协议和EIGRP协议进程）学习到的路由，通过路由选择重分配的方式注入到OSPF域的路由器。一台ASBR路由器可以是位于OSPF域的自主系统内部的任何路由器，它可以是一台内部路由器、骨干路由器或者ABR路由器。

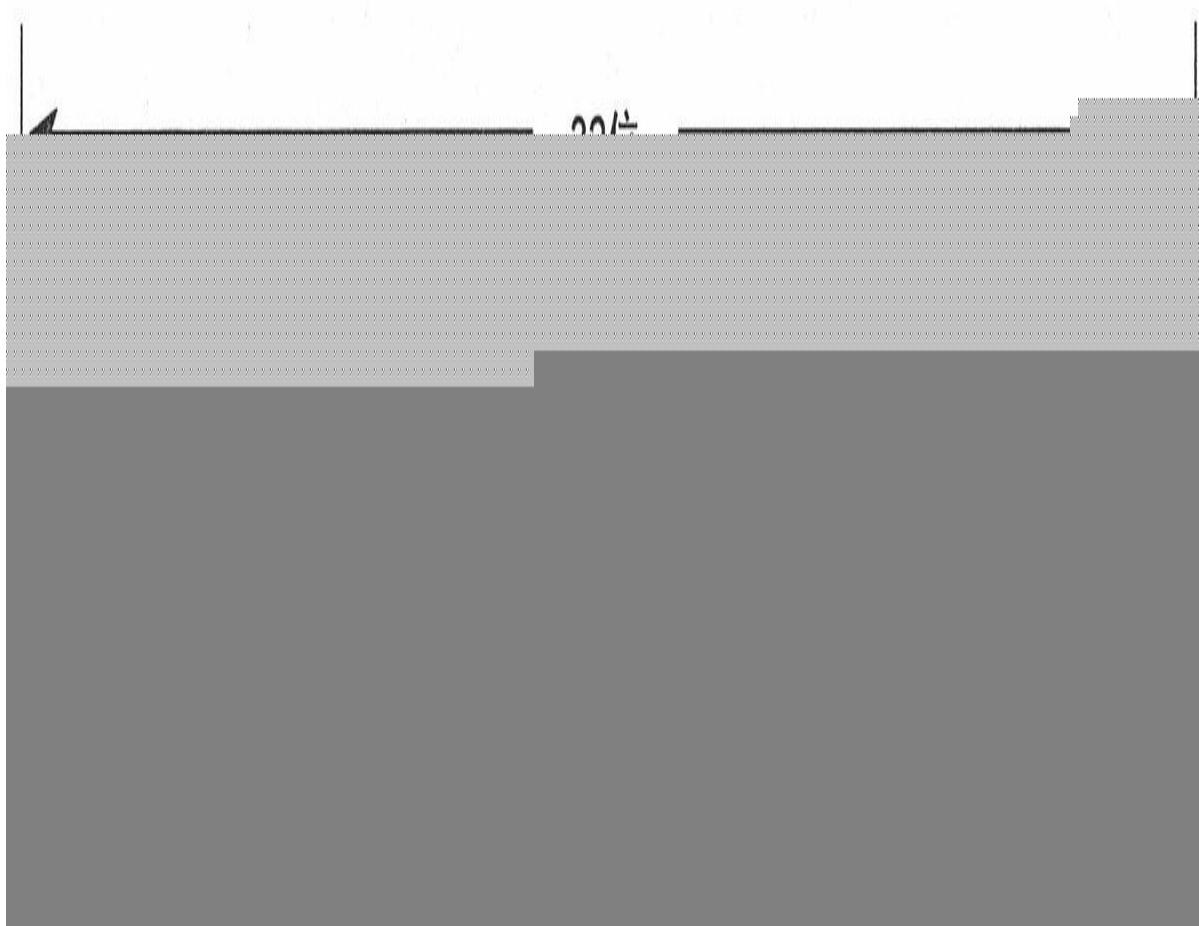


图8-13 所有的**OSPF**路由器都可以被归类到下面**4**种路由器类型之一——内部路由器、骨干路由器、区域边界路由器（**ABR**）或自主系统边界路由器（**ASBR**）。注意前**3**种路由器类型的任何一种都可能成为一台**ASBR**

2. 分段区域

分段区域（partitioned area）是指由于链路失效而使一个区域的一个部分和其他部分隔离开来的情形。如果一个非骨干的区域变成分段区域，并且在这个分段区域的任何一段区域里的所有路由器当中都还能发现一台**ABR**路由器，如图8-14所示，那么这个分段区域将不会产生中断通信服务的情况。骨干区域仅仅会把这个分段区域看成两个单独的区域。但是，从这个分段区域的任何一段区域到另一段区域的域内通信量将变为域间通信量，这些通信量将通过骨干区域而绕开该分段区域。这里要注意，分段区域和孤立区域（isolated area）是不同的，孤立区域没有链路

路径和网络相连。

如果一个骨干区域本身变成了分段区域，那么将会带来更加麻烦的问题。如图8-15中所示，一个分段的骨干区域将把原来的骨干区域隔离成两部分区域，并在这两部分区域上创建两个单独的OSPF域。

如图8-16所示，图中显示了一些更好的区域设计方法。在区域0和区域2之间设计了两条数据链路，这样任何一条链路失效都不会使它们变成分段的区域。但是，区域2也有一个设计缺陷就是如果ABR路由器失效了，那么这个区域就会被孤立。区域3使用了两台ABR路由器，在这里，任何单条链路的失效或单个ABR的失效都不会隔离这个区域的任何部分。

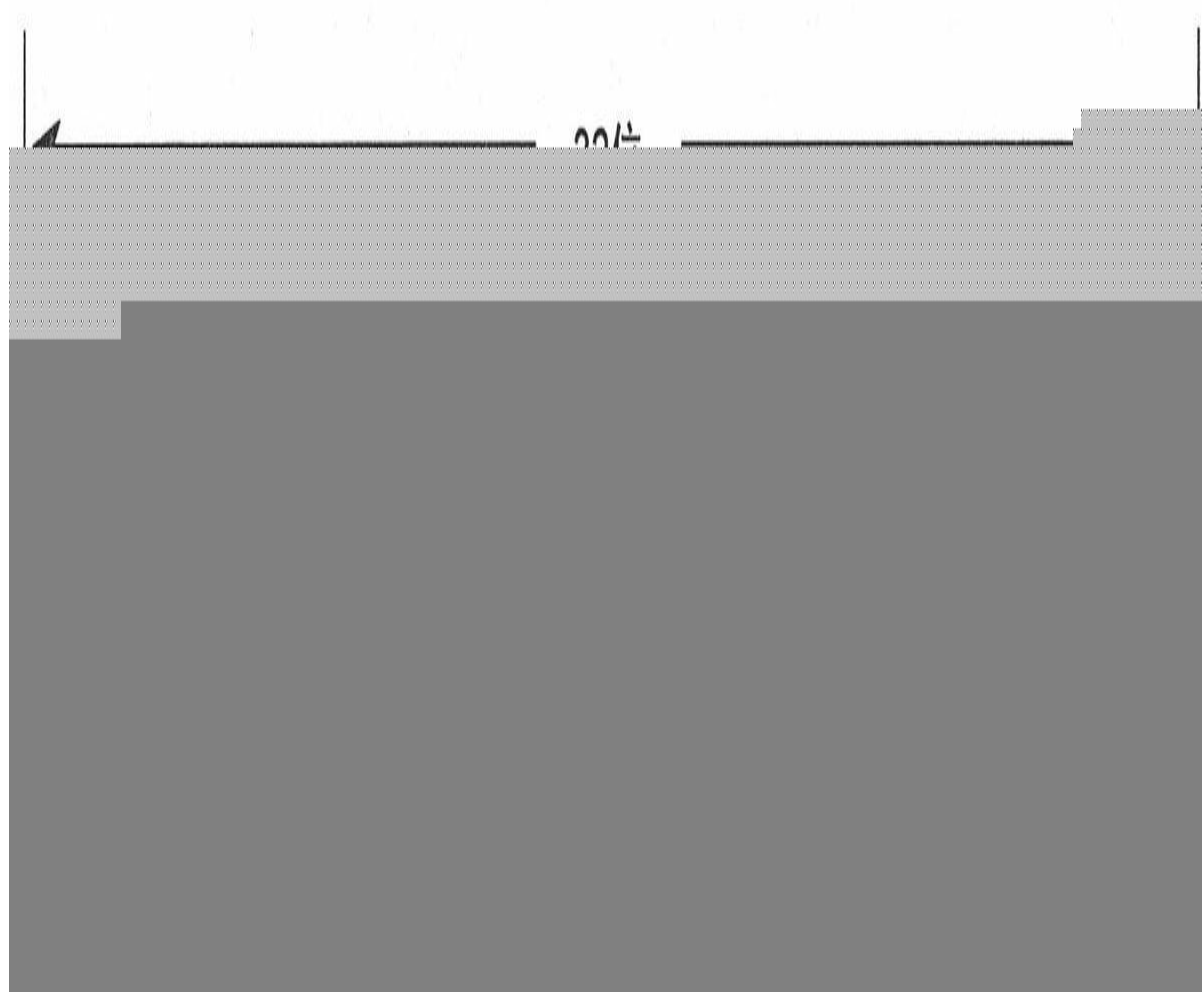


图8-14 （a）区域3通过两台ABR路由器和骨干区域（区域0）相连。
（b）区域3的一条链路失效了将会创建一个分段区域，但是区域3内的

所有路由器都仍然可以到达一台**ABR**路由器。在这种情况下，数据的通信量仍然可以在这个分段区域的两边进行转发

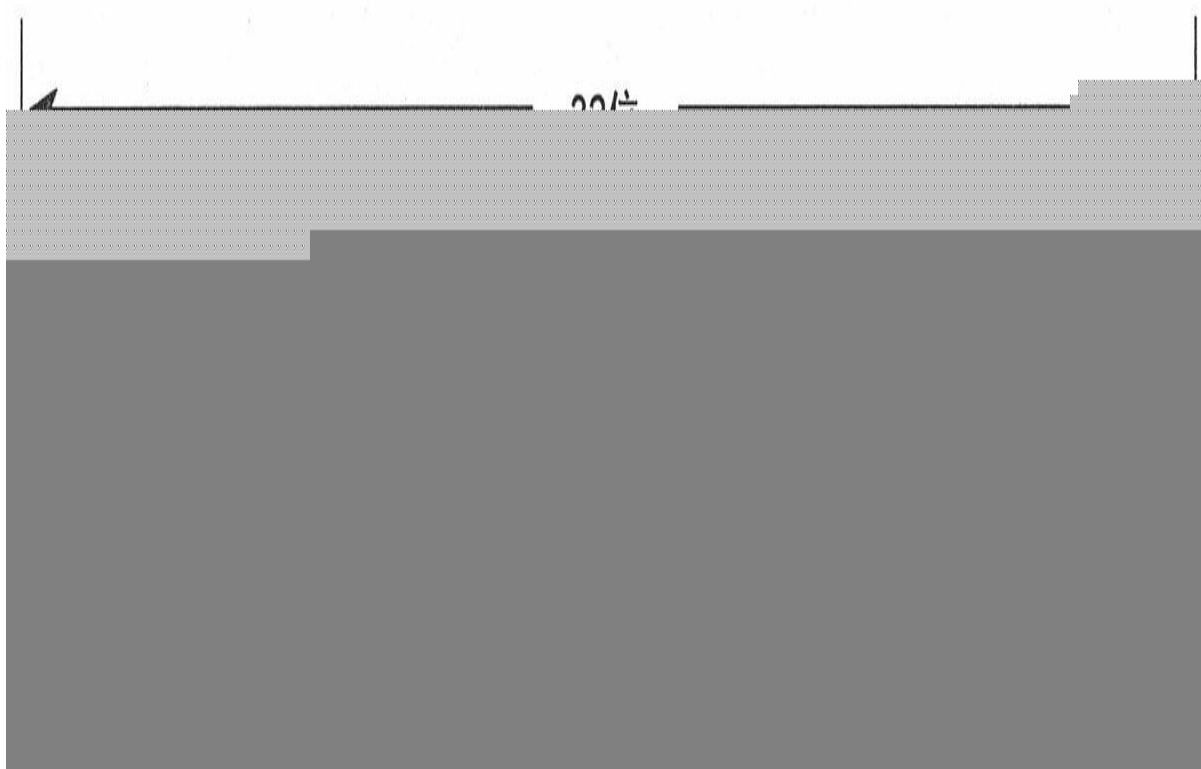


图8-15 如果一个骨干区域变成分段的区域，那么这个分段的骨干区域的每一边和与之相连的区域都将和另外一边的部分隔离开来

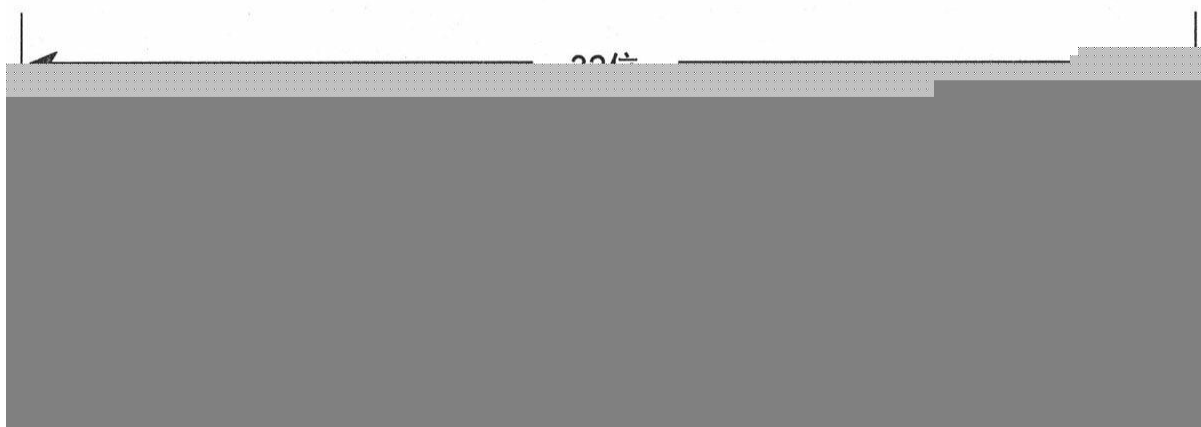


图8-16 在区域0和区域2之间，单条链路的失效不会隔离这个区域。在区域3中，单条链路或单台**ABR**路由器的失效也不会隔离区域3

3. 虚链路

虚链路（virtual link）是指一条通过一个非骨干区域连接到骨干区域的链路。虚链路主要应用于以下几种目的：

（1）通过一个非骨干区域连接一个区域到骨干区域（如图8-17所示）。

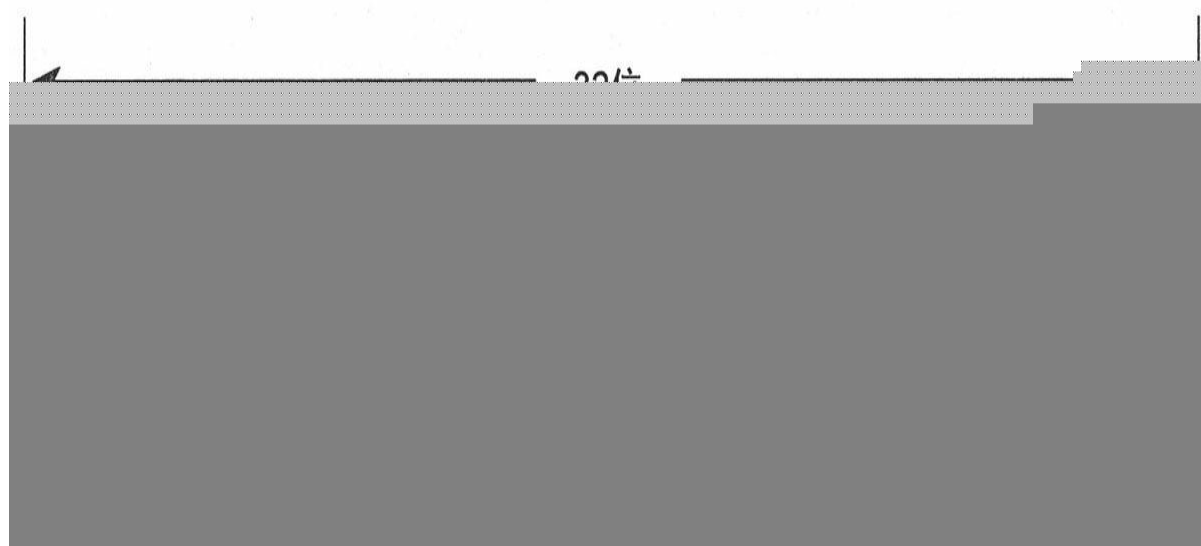


图8-17 一条虚链路用来把区域23经由区域12连接到骨干区域

（2）通过一个非骨干区域连接一个分段的骨干区域两边的部分区域（如图8-18所示）。

在这两个例子中，虚链路和具体的物理链路没有关系。虚链路事实上是一个逻辑通道（tunnel），数据包可以通过选择最优的路径从一端到达另一端。

在配置虚链路的时候，有几条相关的规则，说明如下：

- 虚链路必须配置在两台ABR路由器之间；
- 配置了虚链路所经过的区域必须拥有全部的路由选择信息，这样的区域又被称为传送区域（transit area）；
- 传送区域不能是一个末梢区域。

正如前面所提及的，OSPF协议也把虚链路归类为一个网络类型。更特别的是，虚链路可以看成是在两台ABR路由器之间的一个无编码的——也就是说是无编址地址——链路，并且它是属于骨干区域的。这些ABR路由器之间虽然没有物理的数据链路相连，但是它们可以看作是通通过它们之间的虚链路逻辑上虚拟连接的邻居。在每一个ABR路由器的路由表中，当发现有到达邻居的ABR路由器的路由时，虚链路将转换到完全可操作的点到点接口状态。这条虚链路的代价就是到达它的邻居路由器的路由代价。当接口状态变为点到点状态时，一个邻接关系将通过这条虚链路建立成功。

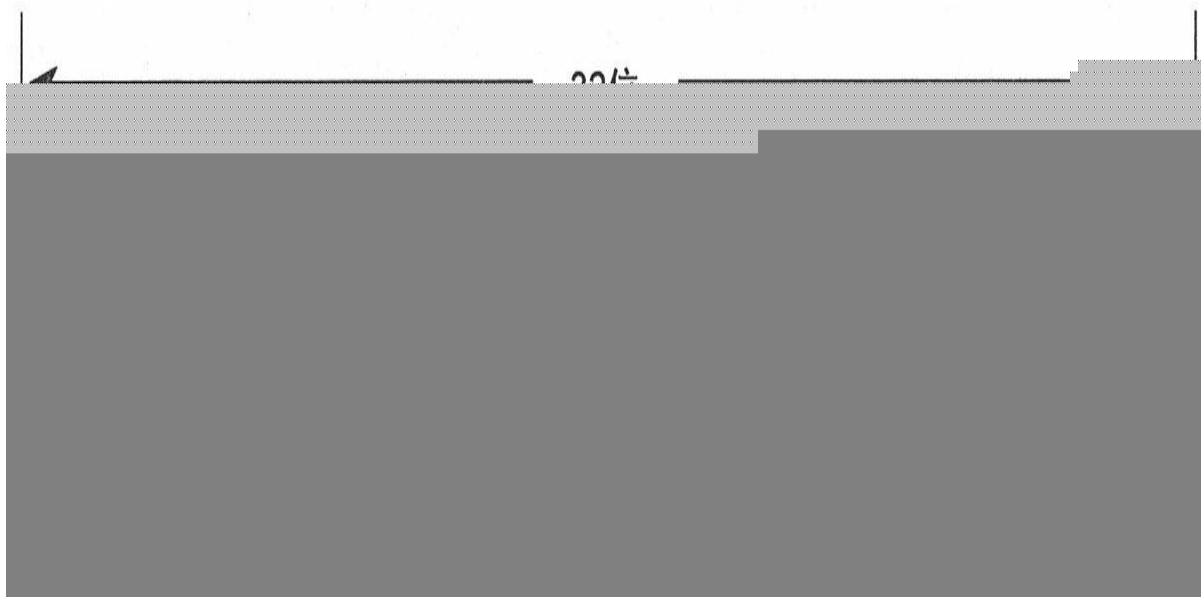


图8-18 一条虚链路穿过一个非骨干区域重新连接一个分段的骨干区域

显然，虚链路的存在增加了网络的复杂程度，而且使故障的排除更加困难。因此，最好避免使用虚链路，而应该在区域上，特别是骨干区域上设计冗余链路来确保防止分段区域的产生。当有两个或多个网络要合并时，预先要制定好充分的计划，以便确保那些没有直连链路到达骨干区域的区域不被遗漏。

如果配置了一条虚链路，设计者应该仅仅把它用来作为修复无法避免的网络拓扑问题的一种临时手段。虚链路可以看作是一个标明网络的某个部分是否需要重新设计的标志。事实上，永久虚链路的存在总是一个设计比较糟糕的网络的标志。

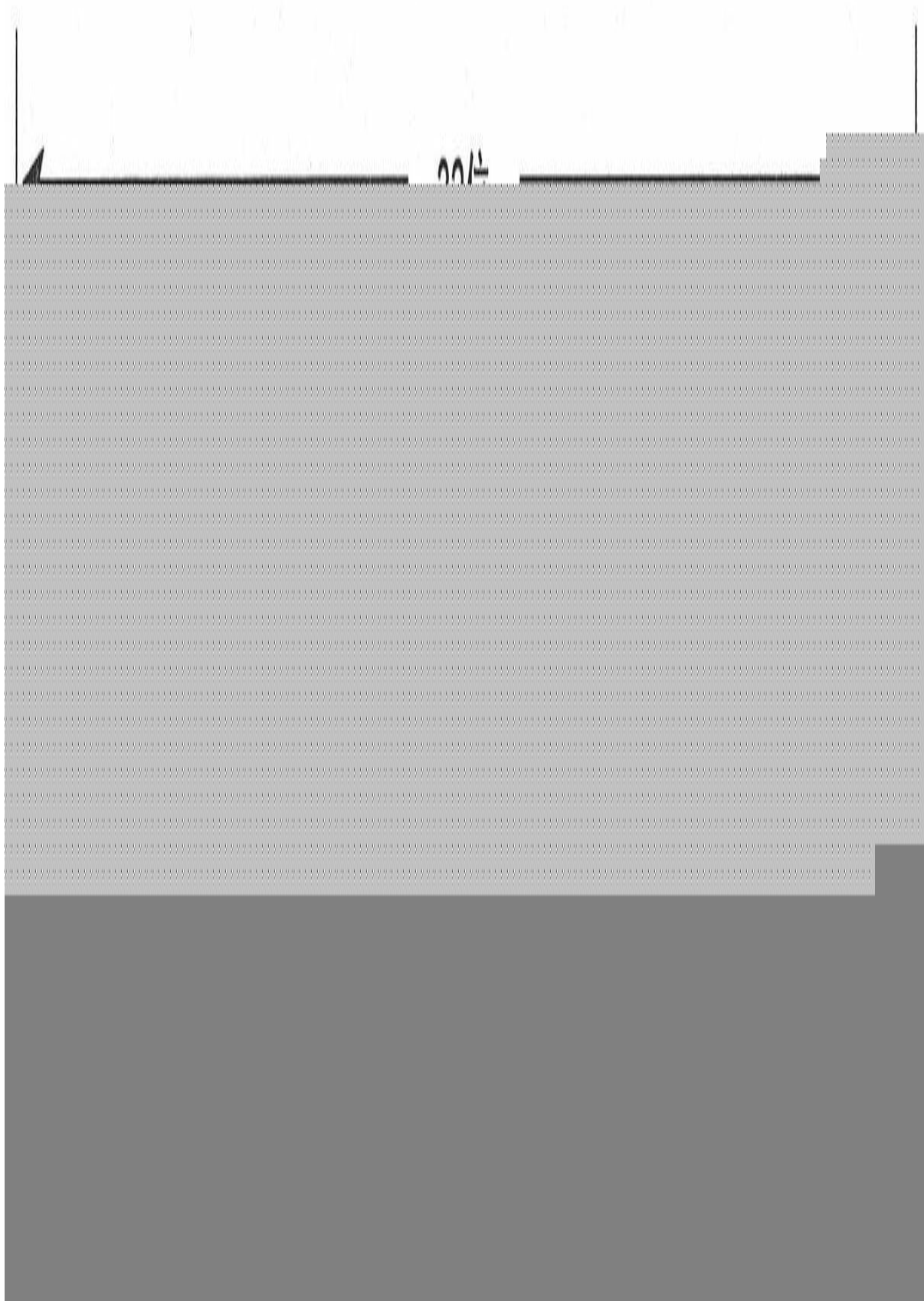
8.1.3 链路状态数据库

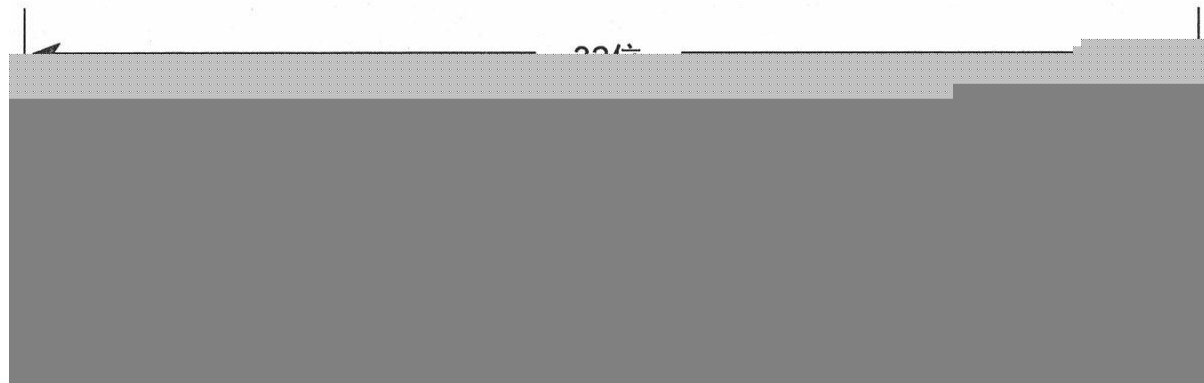
一台路由器中所有有效的LSA都被存放在它的链路状态数据库中。正确的LSA将可以描述一个OSPF区域网络拓扑的结构。因为一个区域中的每一台路由器都要利用这个数据库的信息来计算它自己的最短路径树，因此，所有区域数据库的统一性对于正确的路由选择来说就变得十分重要。

参见示例8-9所示，要观察一个链路状态数据库中的所有LSA的列表可以通过命令**show ip ospf database**来实现。图中所显示的列表并不是数据库中存储的关于每个LSA的全部信息，而仅仅是这些LSA的头部信息。这里要注意，这个数据库如果是包含多个区域的LSA信息的，那么就表明这台路由器是ABR路由器。

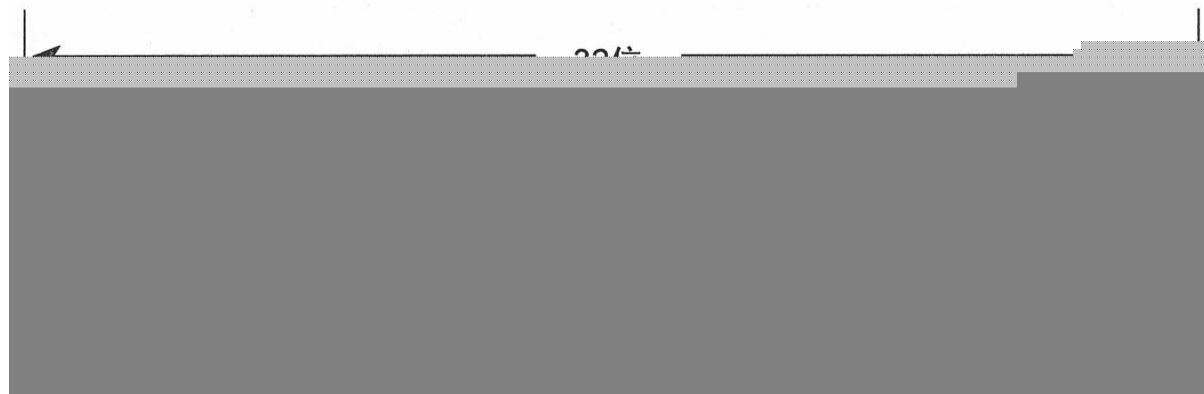
示例8-9中的大多数条目出于简化的目的已经被删除，真实的链路状态数据库包含了 1445个LSA条目和4个区域，参见示例8-10所示。

示例8-9 使用命令**show ip ospf database**来显示一个链路状态数据库中所有LSA的列表





示例8-10 使用命令**show ip ospf database database-summary**来显示一个链路状态数据库当中基于区域和LSA类型分类的LSA通告的数量



正如早前在“可靠的泛洪扩散：序列号、校验和、老化时间”部分所提及的，当LSA通告驻留在路由器的链路状态数据库中的时候，它们的老化时间是增大的。如果这些LSA通告达到了最大生存时间（1h），那么它们将从OSPF域中清除掉。这就意味着，在这里必须有一种机制来防止正常的LSA通告达到最大生存时间而被清除掉。这种机制就是链路状态重刷新（link state refresh）。每隔30min（这个时间称为LSRefreshTime）始发这条LSA通告的路由器就将泛洪扩散这条LSA的一个新拷贝，并将它的序列号增加1，老化时间设置为0。其他的OSPF路由器一旦收到这个新拷贝，就会用这个新拷贝替换该条LSA通告原来的拷贝，并且使这个新拷贝的老化时间开始增加。

虽然链路状态重刷新的机制是用来确保每条LSA通告的活动状态的，但是，它还带来一个额外的好处是，任何一个在路由器的链路状态数据库中可能已经被破坏的LSA通告，都可以被正常LSA通告刷新后的拷贝来替换。

由于每一个LSA通告都与一个独自的重刷新计时器相关联，这意味着每30min, LSA通告的LSRefreshTime将不会一下子都突然超时，从而重新泛洪扩散所有的LSA通告。作为替换做法，重新泛洪扩散将在一个半随机的模式（semi-random pattern）下传播出去。这种方法带来的问题是，每一个单独的LSA通告都会在它增加LSRefreshTime超时的时候被重新泛洪扩散，这使链路带宽的使用没有效率。更新数据包只能传送一些，甚至单个LSA通告。

在IOS软件11.3版本之前，Cisco路由器只选用单个LSRefreshTime和整个链路状态数据库相关联。每隔30min，每台路由器将重刷新它始发的所有LSA通告，而不管这些LSA通告实际的老化时间。虽然这种策略避免了链路带宽使用低效的问题，但是它再次引入了本应解决的独自重刷新计时器的问题。如果一个链路状态数据库很大，那么每隔30min，网络上就会产生一个区域通信量和CPU利用率的高峰。

因此，一种称为LSA组步调（group pacing）的机制，作为LSA独自使用重刷新计时器 和单个统一计时器之间的一种折衷办法。每一个LSA通告都有属于自己的重刷新计时器，但是当它们独自使用的重刷新计时器超时的時候，会引入一个时延来延迟这些LSA通告的泛洪扩散。通过延迟重刷新时间，可以在泛洪扩散之前将更多的LSA通告共同编成一组，从而可以让更新数据包携带更大数量的LSA通告。缺省条件下，一个组步调的间隔时间是240s（4min）。根据IOS软件版本不同，这个间隔时间可以通过命令**timers lsa-group-pacing** 或**timers pacing lsa-group** 来改变。[\[13\]](#)如果链路状态数据库非常大（10000或更多的LSA），那么减小组步调的间隔时间是有好处的；而如果链路状态数据库很小，那么增加组步调的间隔时间会比较有用。在这里，组步调计时器的大小范围是10~1800s。

1. LSA的类型

由于OSPF协议定义了多种路由器的类型，因而定义多种LSA通告的类型也是必要的。例如，一台DR路由器必须通告多路访问链路和所有与这条链路相连的路由器，而其他类型的路由器将不需要通告这种类型的信息。在示例8-9和示例8-10中已经显示了多种类型的LSA通告。每一种LSA通告类型都描述了OSPF网络的一种不同情况。表8-4中列出了LSA通告的类型和标识这些LSA类型的代码。

表8-4

LSA类型

类型代码	描述
1	路由器LSA
2	网络LSA
3	网络汇总LSA
4	ASBR汇总LSA
5	AS外部LSA
6	组成员LSA
7	NSSA部LSA
8	外部属性LSA
9	Opaque LSA（链路本地范围）
10	Opaque LSA（本地区域范围）
11	Opaque LSA（AS范围）

- **路由器LSA（Router LSA）**——每一台路由器都会产生路由器LSA通告（如图8-19所示）。这个最基本的LSA通告列出了路由器所有的链路或接口，并指明了它们的状态和沿每条链路方向出站的代价，以及该链路上所有已知的OSPF邻居。这些LSA通告只会在始发它们的区域内部进行泛洪扩散。通过命令**show ip ospf database router** 可以查看数据库中列出了所有路由器LSA通告。示例8-11显示了这条命令的一个变形，在命令后加了一个变量指定一个路由器ID，从而观察到单台路由器LSA通告的详细信息。在该例及其后面的一些图示中，显示了记录在链路状态数据库中的完整的LSA信息。关于对所有LSA字段的介绍，请参考8.1.7小节中的讲述。

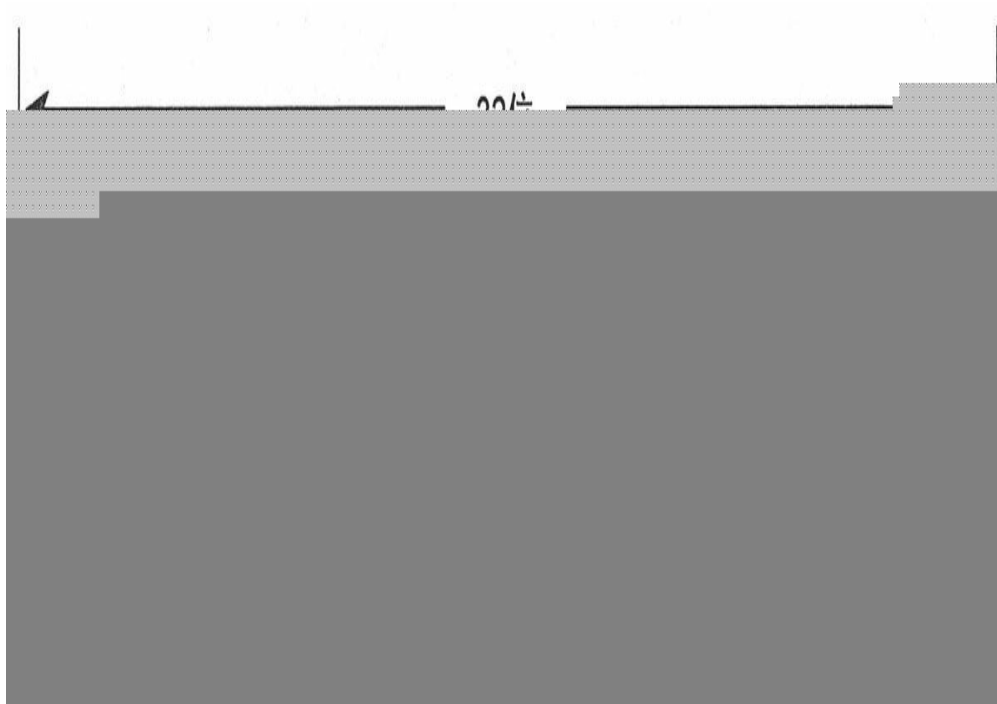
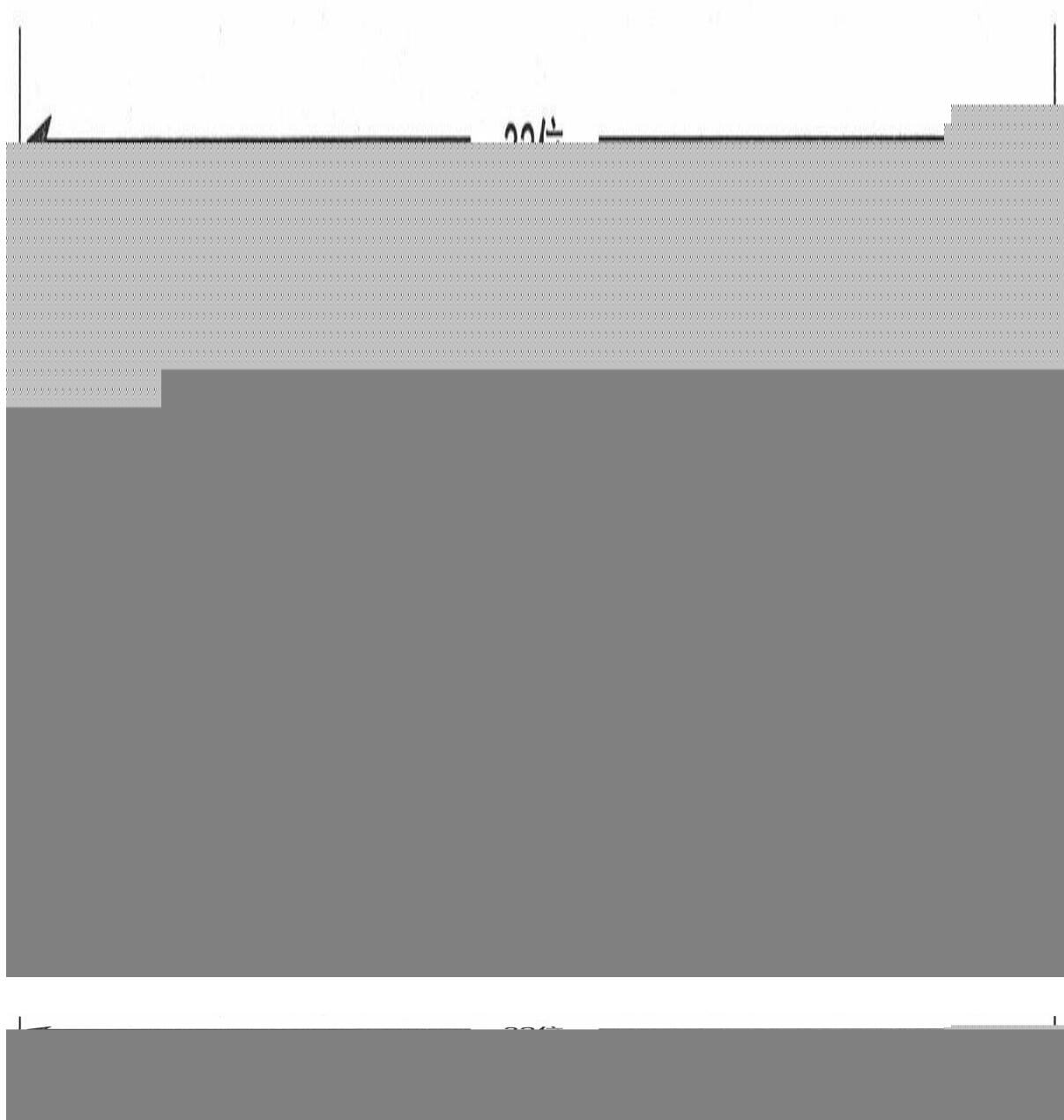


图8-19 路由器LSA描述了所有的路由器接口

示例8-11 通过命令**show ip ospf database router**可以显示出一个链路状态数据库中的路由器LSA通告



请注意，在示例8-11和后面的几个LSA的显示中都有一行是“Routing Bit Set on this LSA”。路由选择位不是LSA本身的一部分；它是IOS软件中用来作内部网络维护的位，表示通过这个LSA通告到目的地的路由是有效的。因此，当你看到“Routing Bit Set on this LSA”时，表示到达这个目的地的路由在路由选择表中。

- **网络LSA（Network LSA）**——每一个多路访问网络中的指定路由器（DR）将会产生网络LSA通告，如图8-20所示。正如前面讨

论的，DR路由器可以看作一个“伪”节点，或是一个虚拟路由器，用来描绘一个多路访问网络和与之相连的所有路由器。从这个角度来看，一条网络LSA通告也可以描绘一个逻辑上的“伪”节点，就像一条路由器LSA通告描绘一个物理上的单台路由器一样。网络LSA通告列出了所有与之相连的路由器，包括DR路由器本身。像路由器LSA一样，网络LSA也仅仅在产生这条网络LSA的区域内部进行泛洪扩散。示例8-12中使用命令**show ip ospf database network** 可以查看一条网络LSA通告的信息。

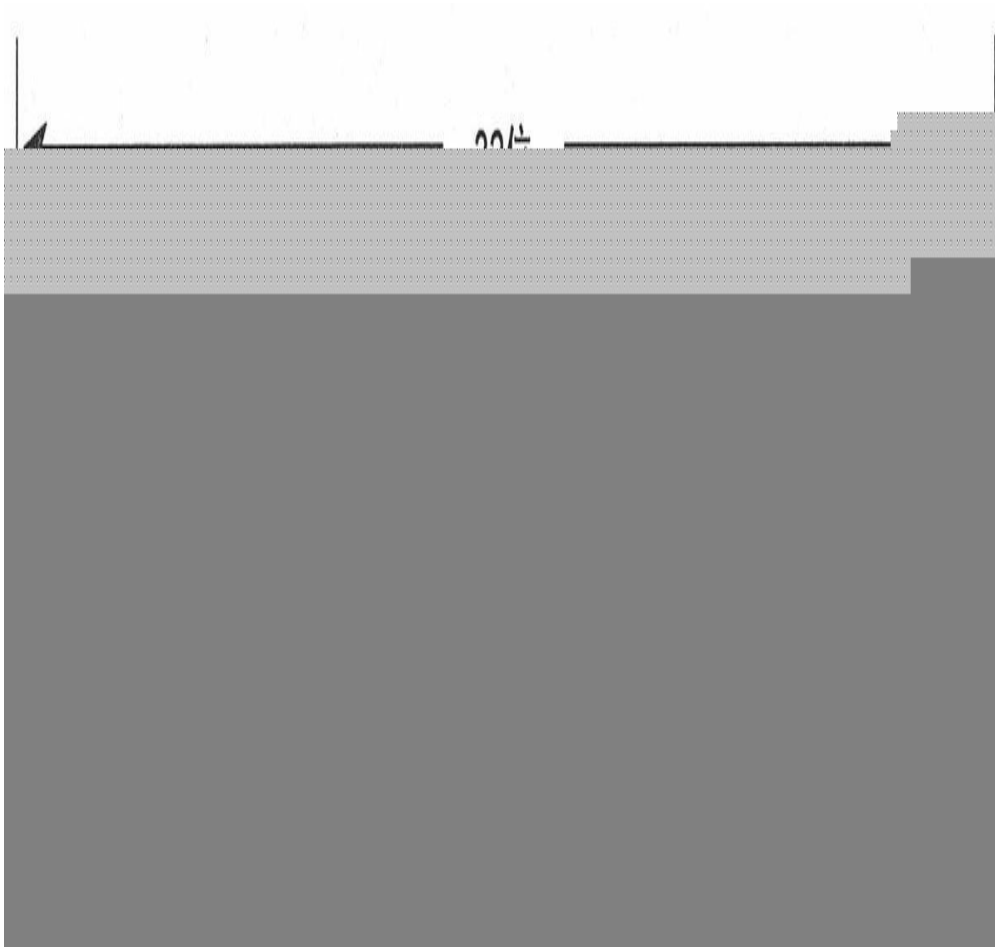
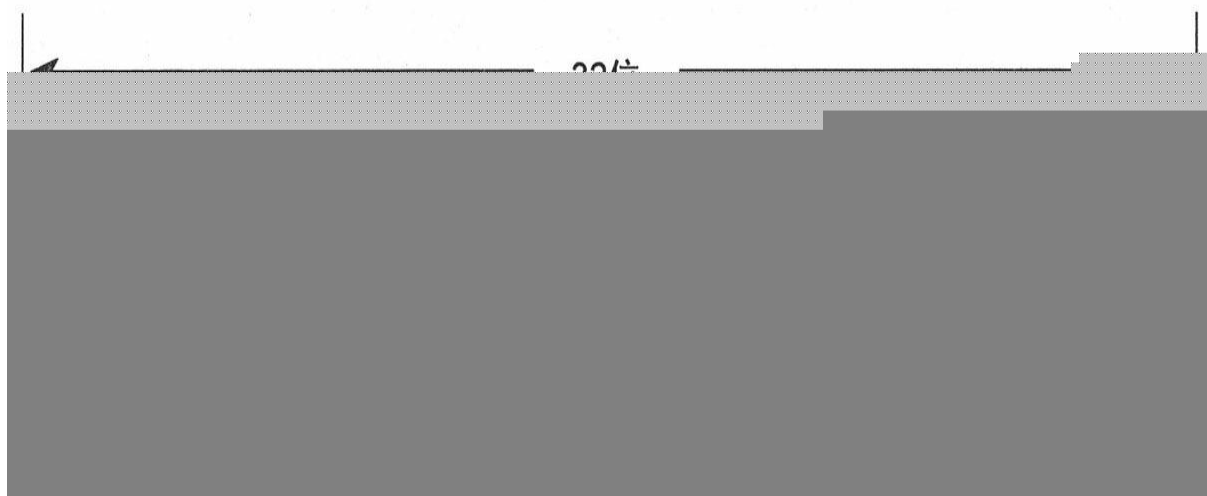
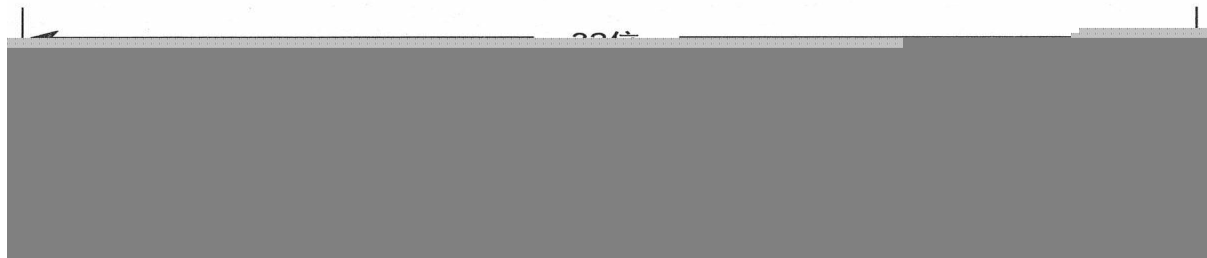


图8-20 DR路由器始发了一条可以表示一个多路访问网络和与之相连的所有路由器的网络LSA通告

示例8-12 网络LSA通告可以通过命令show ip ospf database network来查看



请注意，和路由器LSA不同，网络LSA中没有度量字段。正如本章前面所讲述的，这是因为从LSA中表示的伪节点到任何相连的路由器的代价永远是0。

- **网络汇总LSA（Network Summary LSA）**——是由ABR路由器始发的。ABR路由器将发送网络汇总LSA到一个区域，用来通告该区域外部的目的地址（如图8-21所示）。实际上，这些网络汇总LSA就是ABR路由器告诉在与之相连的区域内的内部路由器它所能到达的目的地址的一种方法。一台ABR路由器也可以通过网络汇总LSA向骨干区域通告与它相连的区域内部的目的地址。在一个区域外部，仍然在一个OSPF自主系统内部的缺省路由也可以通过这种LSA类型来通告。使用命令**show ip ospf database summary**可以显示链路状态数据库中的网络汇总LSA信息，参见示例8-13所示。

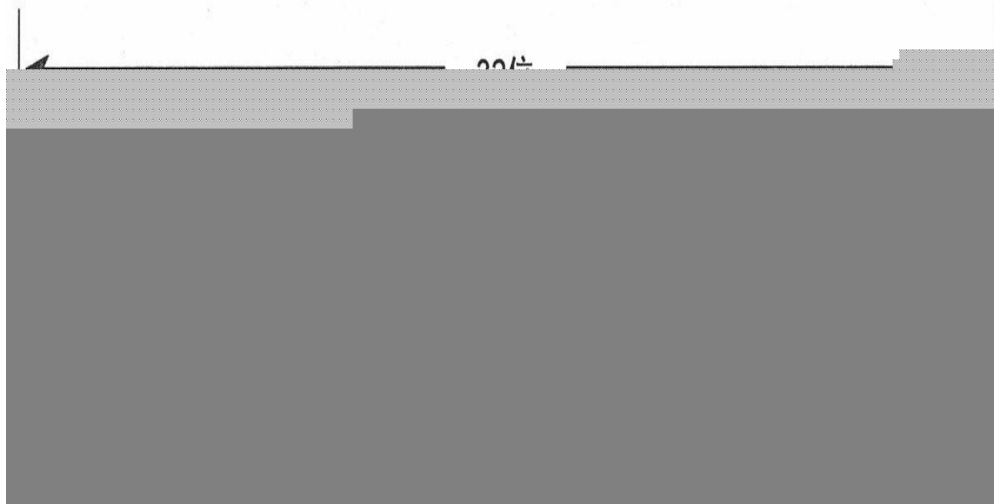
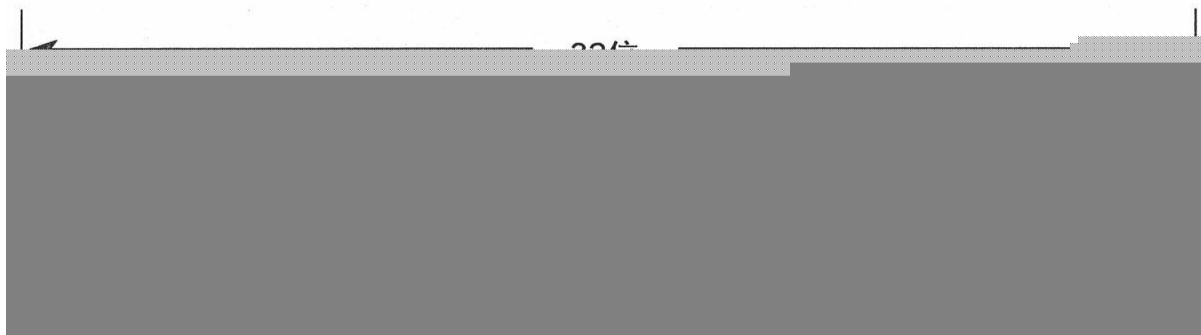
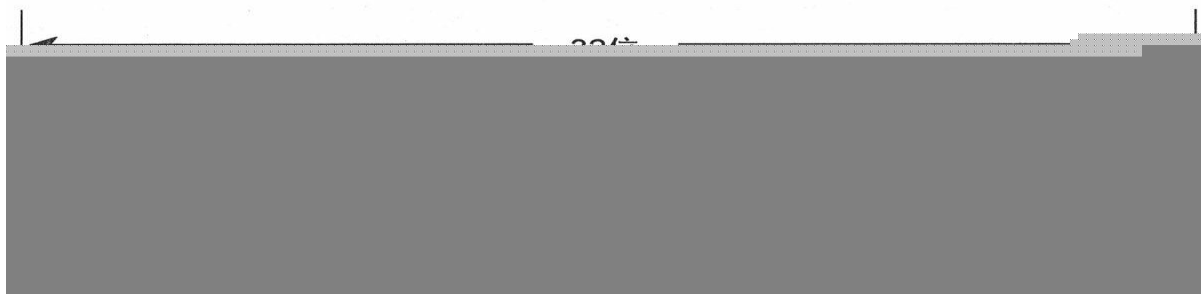


图8-21 一台ABR路由器始发了一条网络汇总LSA来描述域间通信的目的地址

示例8-13 可以通过命令**show ip ospf database summary**来查看网络汇总LSA通告的信息



当一台ABR路由器始发一条网络汇总LSA时，将包括从它本身到正在通告的这条LSA的目的地址所耗费的代价。ABR路由器即使知道它有多条路由可以到达目的地，它也只能为这个目的地始发单条网络汇总LSA通告。因此，如果一台ABR路由器在与它本身相连的区域内有多条路由可

以到达目的地，那么它将只会始发单一的一条网络汇总LSA到骨干区域，而且这条网络汇总LSA是上述多条路由中代价最低的。同样地，如果一台ABR路由器经过骨干区域从其他的ABR路由器收到多条网络汇总LSA，那么这台始发的ABR路由器将会选择这些LSA通告中代价最低的LSA，并且将把这个LSA的最低代价通告给与它相连的非骨干区域。

当其他的路由器从一台ABR路由器收到一条网络汇总LSA通告时，它并不运行SPF算法。相反地，它只是简单地加上从它到那台ABR路由器之间路由的代价，并将这个代价包含在这个LSA通告当中。通过ABR路由器，到达所通告的目的地的路由连同所计算的代价一起被记录进了路由表。这个行为——依赖中间路由器代替确定到达目的地全程路由（full route）的做法——其实是距离矢量协议的行为。因此，虽然在一个区域内部OSPF协议是一个链路状态协议，但是它却使用了距离矢量的算法来查找域间路由。[\[14\]](#)

- **ASBR汇总LSA（ASBR Summary LSA）**——也是由ABR路由器始发的。ASBR汇总LSA除了所通告的目的地是一台ASBR路由器而不是一个网络外，其他的和网络汇总LSA都是一样的，如图8-22所示。使用命令**show ip ospf database asbr-summary** 可以查看ASBR汇总LSA的信息，参见示例8-14所示。这里要注意，在这个图示中，目的地是一个主机地址，并且掩码是0；通过ASBR汇总LSA通告的目的地将总是一个主机地址，因为它是一条到达一台路由器的路由。

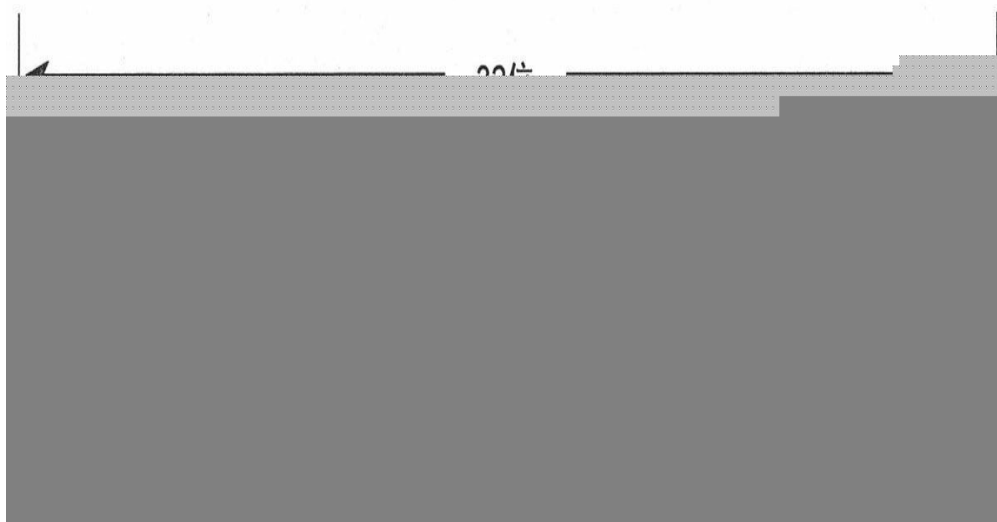
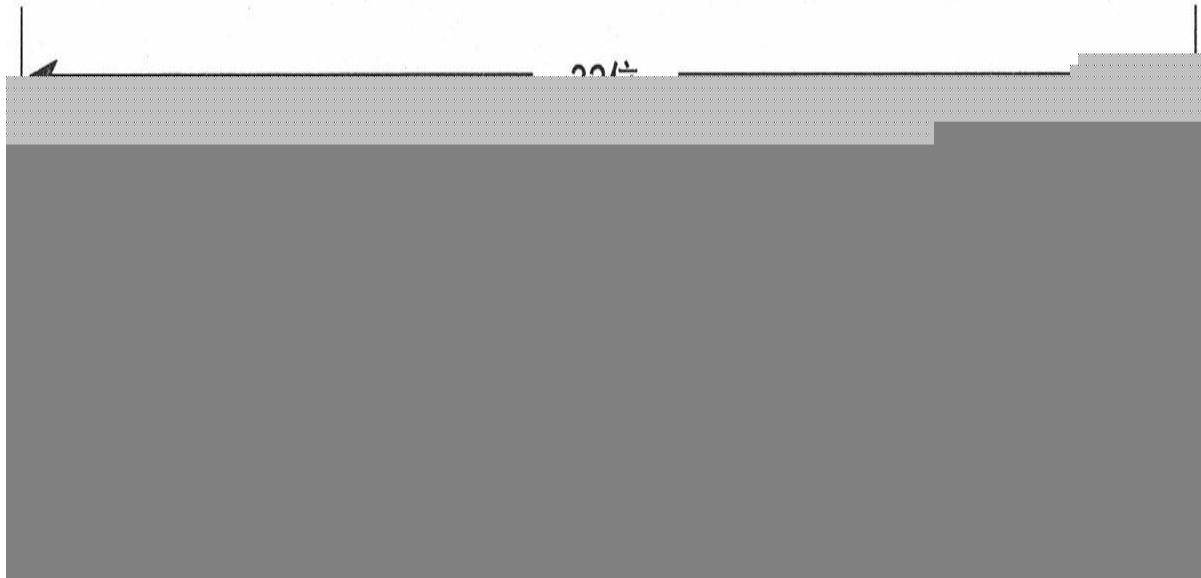


图8-22 ASBR汇总LSA通告到达ASBR路由器的路由

示例8-14 可以通过命令**show ip ospf database asbr-summary**来查看ASBR汇总LSA通告的信息



- 自主系统外部LSA（**Autonomous System External LSA**）——或者称为外部LSA（**External LSA**），是始发于ASBR路由器的，用来通告到达OSPF自主系统外部的目的地或者OSPF自主系统外部的缺省路由[\[15\]](#)的LSA，如图8-23所示。回顾一下示例8-9，可以发现自主系统外部LSA是链路状态数据库中惟一不与具体的区域相关联的LSA通告。外部LSA通告将在整个自主系统中进行泛洪扩散。使用命令**show ip ospf database external** 可以查看AS外部LSA的信息，参见示例8-15所示。

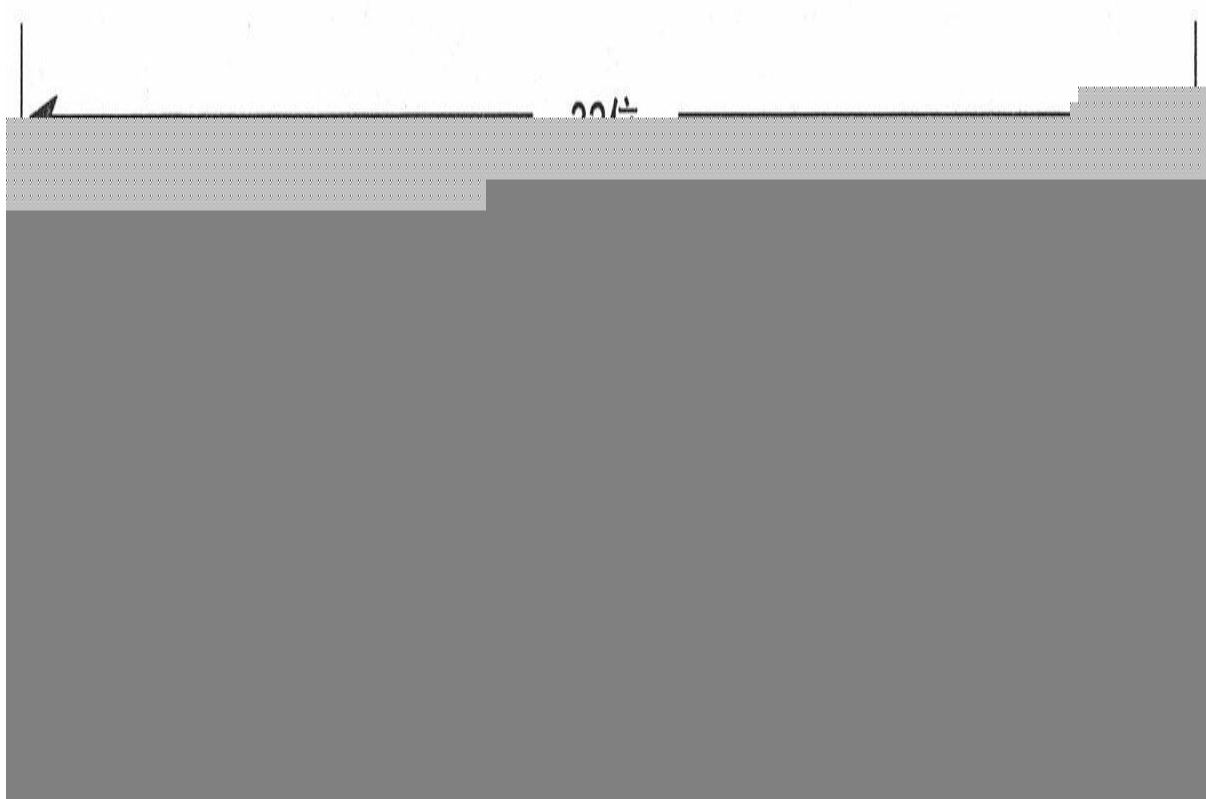


图8-23 自主系统外部LSA用来通告到达OSPF自主系统外部的目的地

示例8-15 可以通过命令**show ip ospf database external**来查看自主系统外部LSA通告的信息

- **组成员LSA（Group Membership LSA）**——是用于OSPF协议的一个增强版本——组播OSPF协议（MOSPF协议）中的。[\[16\]](#) MOSPF协议将数据包从一个单一的源地址转发到多个目的地，或者是一组共享D类组播地址的成员。虽然Cisco路由器支持其他的组播路由选择协议，但是在编写本书时还不支持MOSPF协议。由于这个原因，本书将不介绍MOSPF协议，也不介绍组成员LSA通告。

- **NSSA外部LSA（NSSA External LSA）**——是指在非纯末梢区域（Not-So-Stubby Area, NSSA）内始发于ASBR路由器的LSA通告。NSSA区域将在后面的章节介绍。正像8.1.7小节中所介绍的那样，NSSA外部LSA通告几乎和自主系统外部LSA通告是相同的。只是不像自主系统外部LSA通告那样在整个OSPF自主系统内进行泛洪扩散，NSSA外部LSA通告仅仅在始发这个NSSA外部LSA通告的非纯末梢区域内部进行泛洪扩散。可以通过命令**show ip ospf database nssa-external**来显示NSSA外部LSA通告的信息，参见示例8-16所示。

示例8-16 可以通过命令**show ip ospf database nssa-external**来查看NSSA外部LSA通告的信息



- **外部属性LSA（External Attributes LSA）** ——是被提议作为运行内部BGP协议（iBGP协议）的另一种选择，以便用来传送BGP协议的信息穿过一个OSPF域。这个LSA从来没有在大范围部署过，IOS软件也不支持该LSA。
- **Opaque LSA** ——是由标准的LSA头部后面跟随专用信息组成的一类LSA。[\[17\]](#)这个信息字段可以直接由OSPF协议使用，或者由其他应用分发信息到整个OSPF域间接使用。Opaque LSA类型现在用于对OSPF增加可变的扩展特性，例如在MPLS网络中应用的流量工程参数。

2. 末梢（Stub）区域

一个学习到外部目的地路由信息的ASBR路由器，将通过在整个OSPF自主系统中泛洪扩散自主系统外部LSA来通告那些外部的目的路由信息。在大多数的实际案例中，这些外部LSA通告可能会在每台路由器的链路状态数据库中构成较大百分比的LSA数量。例如，在示例8-10中显示的那个链路状态数据库中有580个LSA（约40%）是外部LSA通告。

如图8-24所示，并不是每一台路由器都需要了解所有外部目的地的信息。不管外部目的地在哪里，在区域2中的路由器都必须发送数据包到ABR路由器，以便到达那台ASBR路由器。在这种情况下，区域2可以被配置成为一个末梢区域。

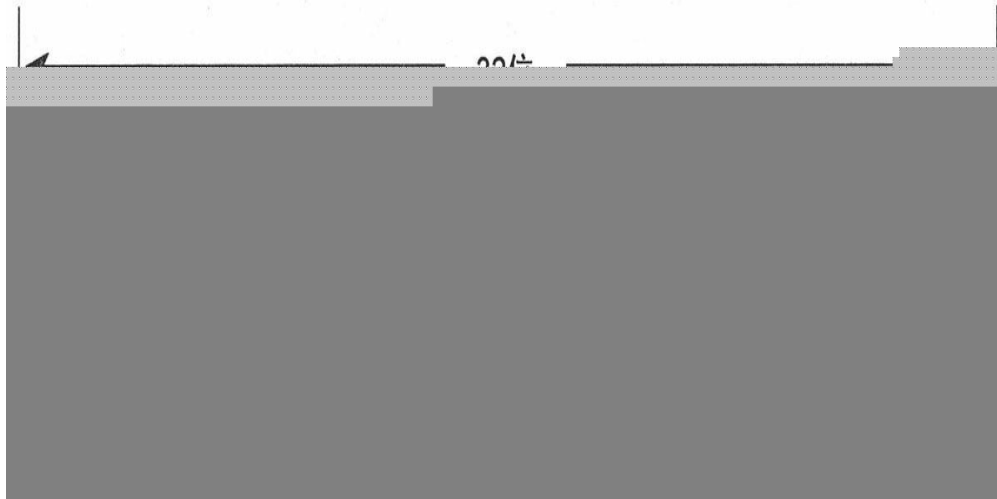


图8-24 可以通过使区域2成为一个末梢区域来节省内存和提高性能

末梢区域是一个不允许AS外部LSA通告在其内部进行泛洪扩散的区域。如果在一个区域里没有学到类型5的LSA通告，那么类型4的LSA通告也是不必要的了，因此这些LSA通告也将被阻塞。位于末梢区域边界的ABR路由器将使用网络汇总LSA向这个区域通告一个简单的缺省路由（目的地址是0.0.0.0）。在区域内部路由器上，所有和域内或域间路由不能匹配的目的地址都将最终匹配这条缺省路由。由于缺省路由是由类型3的LSA通告传送的，因此它将不会被通告到这个区域的外部去。

由于在一个末梢区域里，路由器的链路状态数据库的大小被减小了，因此，这些路由器的性能将得到提高，并且内存也得到节省。当然，在一个含有大量类型5的LSA通告的OSPF域里，这种改进将更加显著。然而，在末梢区域中也有4个限制条件：

- 和所有的区域一样，一个末梢区域内部的所有路由器也必须拥有相同的链路状态数据库。为了确保满足这个条件，所有末梢区域内的路由器都会在它们的Hello数据包中设置一个标志——就是E-bit位，并将它设置为0。这样，这些末梢区域路由器将不接受其他路由器发送的任何E-bit为1的Hello数据包。结果，没有配置成一个末梢区域路由器的任何路由器之间将无法成功建立邻接关系。

- 虚链路不能在一个末梢区域内进行配置，也不能穿过一个末梢区域。
- 末梢区域内的路由器不能是ASBR路由器。这个限制条件是很容易直观地理解的，因为ASBR路由器会产生类型5的LSA通告，而在一个末梢区域内不能存在类型5的LSA通告。
- 一个末梢区域可以拥有多台ABR路由器，但是因为缺省路由的原因，区域内部路由器将不能确定哪一台路由器才是到达ASBR路由器的最优网关。

（1）完全末梢区域

如果通过阻塞类型5和类型4的LSA传播到一个区域的方法来节省内存的话，那么要是能够把类型3的LSA也阻塞掉，不是可以节省更多的内存吗？对于这个问题，Cisco借助于末梢区域的概念提出了称为完全末梢区域的概念。

完全末梢区域（totally stubby area）不仅使用缺省路由到达OSPF自主系统外部的目的地址，而且使用缺省路由到达这个区域外部的所有目的地址。一个完全末梢区域的ABR将不仅阻塞AS外部LSA，而且阻塞所有的汇总LSA——除了通告缺省路由的那一条类型3的LSA。

（2）非纯末梢区域

在图8-25中，带有一些末梢网络的某台路由器必须通过区域2的其中一台路由器和OSPF网络相连。但是，该路由器仅支持RIP协议，因此，区域2的那台路由器将同时运行RIP协议和OSPF协议，并利用路由重新分配的方法把该路由器的那些末梢网络注入到OSPF域。不幸的是，这种配置将使区域2中的那台路由器成为一台ASBR路由器，因此，区域2也就不再是一个末梢区域了。

在这里，RIP协议的宣告者并不需要学习OSPF域的路由，而只需要有一条缺省路由指向那台区域2的路由器就足够了。但是，OSPF域内的路由器为了能够正确地将数据包转发到RIP路由器的那些目的地址，它们必须学习到和RIP路由器相连的目的网络。

非纯末梢区域（Not-So-Stubby-Area, NSSA）[\[18\]](#)允许外部路由通告到

OSPF自主系统内部，而同时保留自主系统其余部分的末梢区域特征。为了做到这一点，在NSSA区域内的ASBR将始发类型7的LSA用来通告那些外部的目的网络。这些NSSA外部LSA将在整个NSSA区域中进行泛洪扩散，但是会在ABR路由器的地方被阻塞。

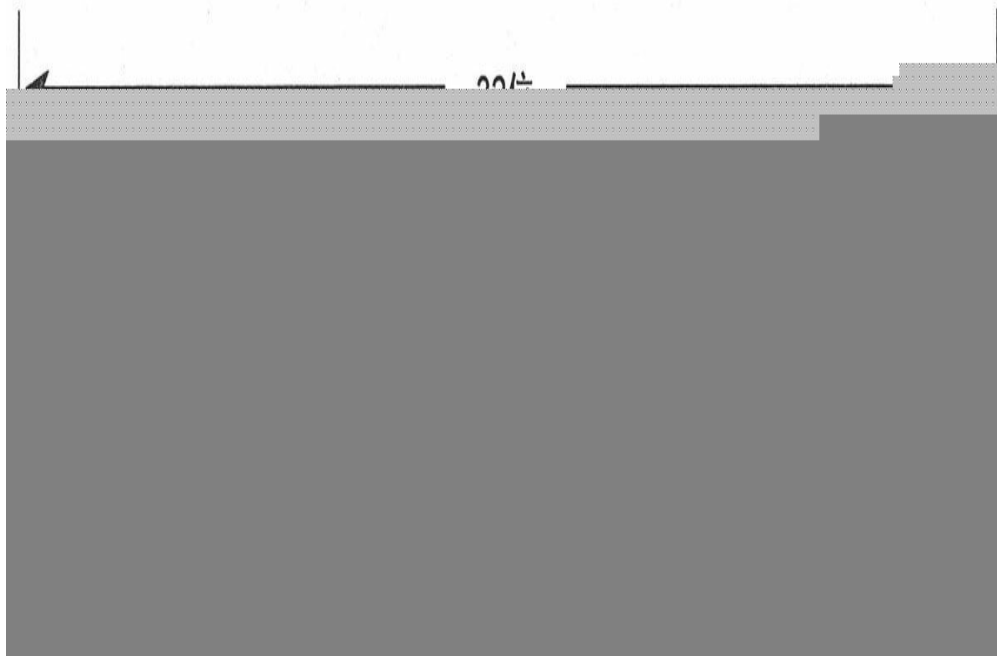


图8-25 因为有一些**OSPF**域外部的目的网络必须在**区域2**的路由器上通过路由重新分配的方式注入到**OSPF**域中，因此**区域2**就不再满足末梢区域的条件了

NSSA外部LSA在它的头部有一个称为P-bit位的标志。NSSA ASBR路由器可以设置或清除这个P-bit位。如果一台NSSA ABR路由器收到一条P-bit设置为1的类型7的LSA数据包，那么它将把这条类型7的LSA转换成类型5的LSA，并且将这条LSA泛洪扩散到其他的区域中去（参见图8-26）。如果这个P-bit位被设置为0，那么将不会转换这条类型7的LSA，而且这条类型7的LSA携带的目的地址也不能通告到这个NSSA区域的外部。这个选项允许我们设计一个NSSA，使那个区域学到的外部目的地址仅仅被那个区域知晓。

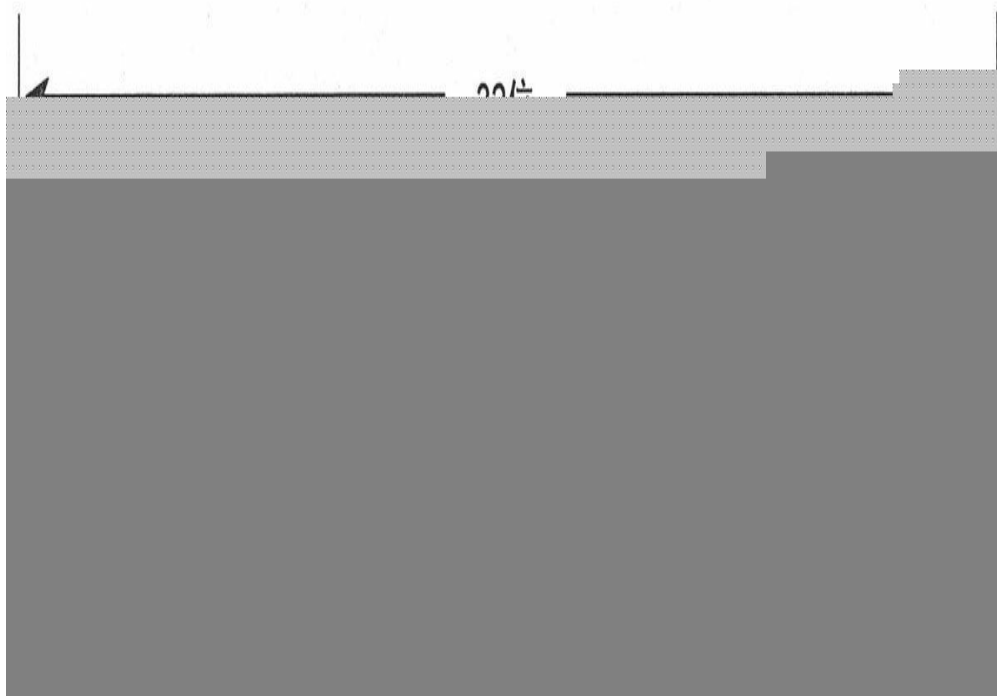


图8-26 在NSSA区域内的ASBR路由器将会始发NSSA外部LSA。如果一条NSSA外部LSA的P-bit位设置了，那么ABR路由器将会把这条NSSA外部LSA转换为一条AS外部LSA

NSSA区域是在IOS软件11.2版及其以后的版本才被支持的。

表8-5总结了每一种区域内允许泛洪扩散的LSA类型。

表8-5 每一种区域内允许泛洪扩散的LSA类型

8.1.4 路由表

根据链路状态数据库中的LSA信息，路由器使用Dijkstra算法来计算一棵最短路径树。第4章已经比较详细地讲述了Dijkstra算法，如果需要查看关于OSPF协议计算SPF树的完整描述，请参考RFC 2328中的第16.1节。

OSPF协议是基于路由器的每一个接口指定的度量值来决定最短路径的，这里的度量值指的是接口指定的代价（cost）。一条路由的代价是指沿着到达目的网络的路由路径上所有出站接口的代价之和。RFC 2328没有专门为代价指定任何值。Cisco路由器使用 $10^8 / \text{BW}$ 作为缺省的OSPF代价，这里的BW是指在路由器接口上配置的带宽， 10^8 是一个参考带宽。正如前面所讲述的，缺省的参考带宽可以通过命令**auto-cost reference-bandwidth**来更改。如果计算所得的结果是分数的话就采用四舍五入的方法取整数值。表8-6中显示了一些典型接口根据这种计算方式得出的缺省代价。

使用命令**ip ospf cost**可以替换缺省的自动进行的代价计算，这条命令可以给某个接口分配一个固定的代价。例如，在一个具有相同骨干链路速率的大型网络中，可以根据视距或线缆/光缆距离来分配链路代价。LSA在16位的字段中记录代价，因此一个接口的总计代价范围可以是1~65535。

表8-6 Cisco路由器的缺省接口代价

接口类型	代价（ $10^8 / \text{BW}$ ）
FDDI、快速以太网，任何接口>100Mbit/s	1
HSSI（45Mbit/s）	2
16Mbit/s令牌环	6
以太网（10Mbit/s）	10
4Mbit/s令牌环	25
T1（1.544Mbit/s）	64
DSO（64kbit/s）*	1562
56kbit/s*	1785
Tunnel（9kbit/s）	11111

* 假定串行接口的缺省带宽已被改变。

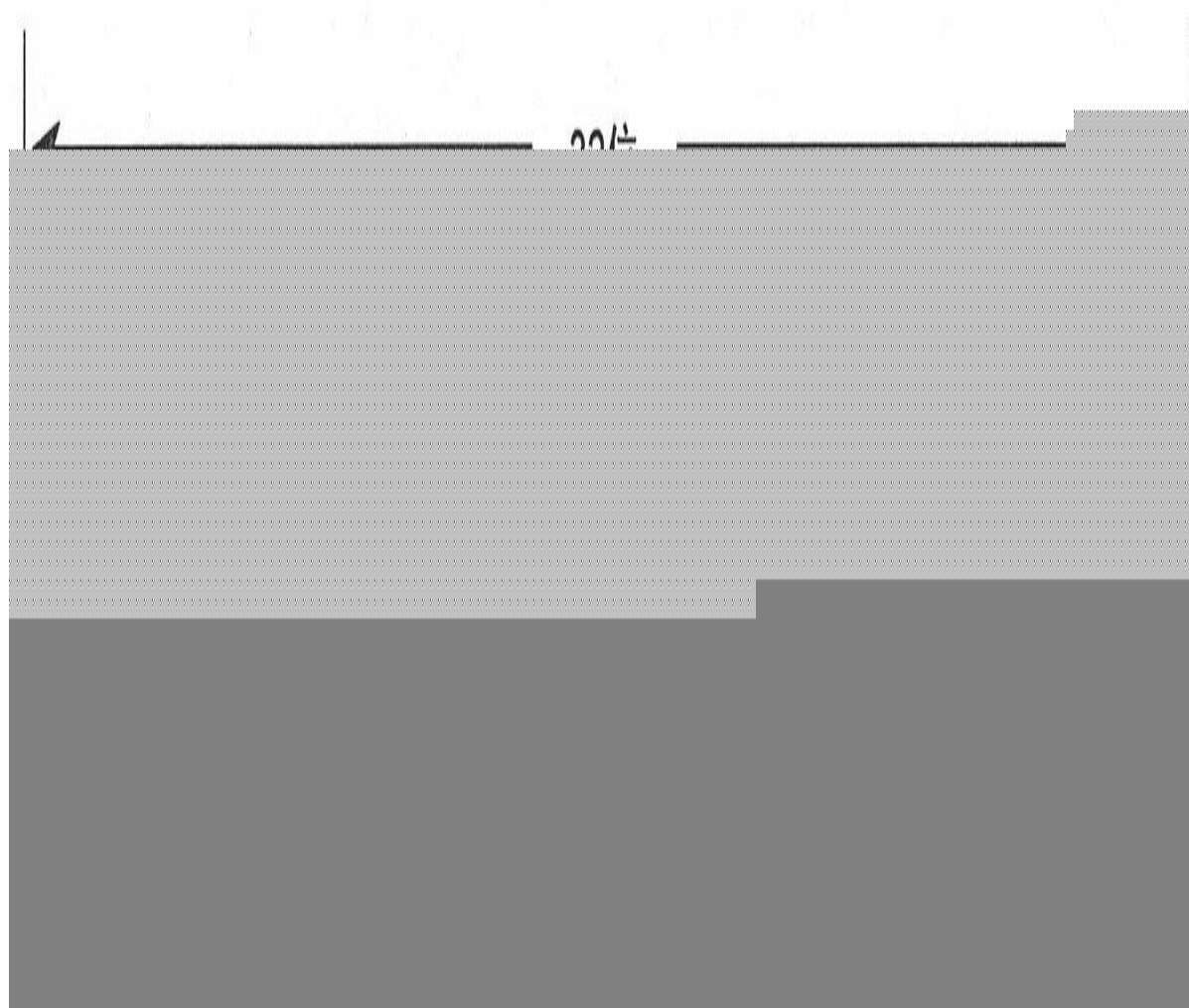
1. 目的类型（Destination Type）

每一个路由条目都可以被归类到目的类型中去。目的类型可以是网络也

可以是路由器。

网络条目（**network entries**）是数据包所要转发的目的网络地址。这些网络条目就是记录到路由表中的目的网络地址，参见示例8-17所示。

示例8-17 在路由表中的**OSPF**条目是网络目的类型



路由器条目（**router entries**）是到达ABR和ASBR路由器的路由。如果一台路由器需要发送数据包到一个区域外的目的地，那么它就必须知道如何到达一台ABR路由器；同样的，如果一台路由器需要发送数据包到一个OSPF域外部的目的地，那么它就必须知道如何到达一台ASBR路由器。路由器条目就包含了这个信息，并且路由器把路由器条目放在了一个单独的内部路由表中。这个路由表可以通过命令**show ip ospf border-routers** 来查看，参见示例8-18所示。

正如示例8-18所显示的，这个内部的路由表看上去和其他普通路由表十分相似：也包含目的地、度量值、下一跳地址和出口接口。这里所不同的是，在这个内部的路由表中的所有目的地都是ABR和ASBR的路由器ID。每一个条目都被打上区域内（i）或区域间（I）标志，用来表明这个条目的目的地是一台ABR，或是一台ASBR，或两者都是。所在的区域也被记录了，以便用来进行SPF算法的迭代查找。

示例8-18 路由器条目放置在一个和网络条目相分开的内部表中，用来表示到达ABR和ASBR路由器的路由

The image contains two screenshots of a network router's internal routing table. The top screenshot shows a table with columns for destination, metric, next hop, and interface. The bottom screenshot shows a similar table with an additional column for area ID.

目的地	度量值	下一跳地址	出口接口
10.1.1.1	10	10.1.1.1	10/1
10.1.1.2	10	10.1.1.2	10/1
10.1.1.3	10	10.1.1.3	10/1
10.1.1.4	10	10.1.1.4	10/1
10.1.1.5	10	10.1.1.5	10/1
10.1.1.6	10	10.1.1.6	10/1
10.1.1.7	10	10.1.1.7	10/1
10.1.1.8	10	10.1.1.8	10/1
10.1.1.9	10	10.1.1.9	10/1
10.1.1.10	10	10.1.1.10	10/1
10.1.1.11	10	10.1.1.11	10/1
10.1.1.12	10	10.1.1.12	10/1
10.1.1.13	10	10.1.1.13	10/1
10.1.1.14	10	10.1.1.14	10/1
10.1.1.15	10	10.1.1.15	10/1
10.1.1.16	10	10.1.1.16	10/1
10.1.1.17	10	10.1.1.17	10/1
10.1.1.18	10	10.1.1.18	10/1
10.1.1.19	10	10.1.1.19	10/1
10.1.1.20	10	10.1.1.20	10/1
10.1.1.21	10	10.1.1.21	10/1
10.1.1.22	10	10.1.1.22	10/1
10.1.1.23	10	10.1.1.23	10/1
10.1.1.24	10	10.1.1.24	10/1
10.1.1.25	10	10.1.1.25	10/1
10.1.1.26	10	10.1.1.26	10/1
10.1.1.27	10	10.1.1.27	10/1
10.1.1.28	10	10.1.1.28	10/1
10.1.1.29	10	10.1.1.29	10/1
10.1.1.30	10	10.1.1.30	10/1
10.1.1.31	10	10.1.1.31	10/1
10.1.1.32	10	10.1.1.32	10/1
10.1.1.33	10	10.1.1.33	10/1
10.1.1.34	10	10.1.1.34	10/1
10.1.1.35	10	10.1.1.35	10/1
10.1.1.36	10	10.1.1.36	10/1
10.1.1.37	10	10.1.1.37	10/1
10.1.1.38	10	10.1.1.38	10/1
10.1.1.39	10	10.1.1.39	10/1
10.1.1.40	10	10.1.1.40	10/1
10.1.1.41	10	10.1.1.41	10/1
10.1.1.42	10	10.1.1.42	10/1
10.1.1.43	10	10.1.1.43	10/1
10.1.1.44	10	10.1.1.44	10/1
10.1.1.45	10	10.1.1.45	10/1
10.1.1.46	10	10.1.1.46	10/1
10.1.1.47	10	10.1.1.47	10/1
10.1.1.48	10	10.1.1.48	10/1
10.1.1.49	10	10.1.1.49	10/1
10.1.1.50	10	10.1.1.50	10/1
10.1.1.51	10	10.1.1.51	10/1
10.1.1.52	10	10.1.1.52	10/1
10.1.1.53	10	10.1.1.53	10/1
10.1.1.54	10	10.1.1.54	10/1
10.1.1.55	10	10.1.1.55	10/1
10.1.1.56	10	10.1.1.56	10/1
10.1.1.57	10	10.1.1.57	10/1
10.1.1.58	10	10.1.1.58	10/1
10.1.1.59	10	10.1.1.59	10/1
10.1.1.60	10	10.1.1.60	10/1
10.1.1.61	10	10.1.1.61	10/1
10.1.1.62	10	10.1.1.62	10/1
10.1.1.63	10	10.1.1.63	10/1
10.1.1.64	10	10.1.1.64	10/1
10.1.1.65	10	10.1.1.65	10/1
10.1.1.66	10	10.1.1.66	10/1
10.1.1.67	10	10.1.1.67	10/1
10.1.1.68	10	10.1.1.68	10/1
10.1.1.69	10	10.1.1.69	10/1
10.1.1.70	10	10.1.1.70	10/1
10.1.1.71	10	10.1.1.71	10/1
10.1.1.72	10	10.1.1.72	10/1
10.1.1.73	10	10.1.1.73	10/1
10.1.1.74	10	10.1.1.74	10/1
10.1.1.75	10	10.1.1.75	10/1
10.1.1.76	10	10.1.1.76	10/1
10.1.1.77	10	10.1.1.77	10/1
10.1.1.78	10	10.1.1.78	10/1
10.1.1.79	10	10.1.1.79	10/1
10.1.1.80	10	10.1.1.80	10/1
10.1.1.81	10	10.1.1.81	10/1
10.1.1.82	10	10.1.1.82	10/1
10.1.1.83	10	10.1.1.83	10/1
10.1.1.84	10	10.1.1.84	10/1
10.1.1.85	10	10.1.1.85	10/1
10.1.1.86	10	10.1.1.86	10/1
10.1.1.87	10	10.1.1.87	10/1
10.1.1.88	10	10.1.1.88	10/1
10.1.1.89	10	10.1.1.89	10/1
10.1.1.90	10	10.1.1.90	10/1
10.1.1.91	10	10.1.1.91	10/1
10.1.1.92	10	10.1.1.92	10/1
10.1.1.93	10	10.1.1.93	10/1
10.1.1.94	10	10.1.1.94	10/1
10.1.1.95	10	10.1.1.95	10/1
10.1.1.96	10	10.1.1.96	10/1
10.1.1.97	10	10.1.1.97	10/1
10.1.1.98	10	10.1.1.98	10/1
10.1.1.99	10	10.1.1.99	10/1
10.1.1.100	10	10.1.1.100	10/1

2. 路径类型

每一条到达一个网络目的地的路由都可以被归类到4种路径类型中的一种。这些路径类型（path type）是区域内路径、区域间路径、类型1的外部路径和类型2的外部路径。

- 区域内路径（**Intra-area path**）——是指在路由器所在的区域

内就可以到达目的地的路径。

- 区域间路径（**Inter-area path**）——是指目的地在其他区域但是还在OSPF自主系统内的路径。在示例8-17中，打上了IA标志的条目就是区域间路径，它总是至少通过一台ABR路由器。
- 类型1的外部路径（**Type 1 external path, E1**）——是指目的地在OSPF自主系统外部的路径，在示例8-17中表示为E1。当一条外部路由重新分配到任何一个自主系统时，它都必须指定一个对该自主系统中的路由选择协议有意义的度量值。在OSPF协议里，ASBR路由器的责任是要给通告的外部路由指定一个代价值。对于类型1的外部路径来说，这个代价值是这条路由的外部代价加上到达ASBR路由器的路径代价之和。关于配置一台ASBR路由器使用E1类型的度量来通告一条外部路由（路由重新分配）的介绍将在第11章中讲述。
- 类型2的外部路径（**Type 2 external path, E2**）——也是指目的地在OSPF自主系统外部的路径，但是在计算外部路由的度量时不再计入到达ASBR路由器的路径代价。

E1和E2类型的路由给网络管理员提供了一个选项，这个选项是：选择计算到ASBR的内部代价重要，还是只选择外部路由的外部代价，而忽略到达ASBR的内部代价更重要。例如，“热土豆（hot potato）”路由选择（在网络最近的出口地点把到外部目的地址的数据包转发出该网络）通常要求E1度量，而如果你希望将数据包从到达它们的外部目的地址最近的地点转发出去，就应该使用E2度量。OSPF外部路由在缺省条件下是E2类型路径。

在图8-27中，路由器A有两条到达外部目的网络10.1.2.0的路径。如果目的地址通过E1类型来通告，那么路径A-B-D的代价是35（5+20+10），这条路径将比代价为50（30+10+10）的路径A-C-D优先。如果目的地址通过E2类型来通告，那么到达ASBR路由器的那两条内部路径的代价将被忽略。在这个实例中，路径A-B-D的代价是30（20+10），而路径A-C-D的代价为20（10+10）。显然，后者是优先路径。



图8-27 如果使用**E1**类型的度量来通告到达外部网络**10.1.2.0**的路由，那么路由器**A**将选择路由器**B**作为最近的**ASBR**。如果使用**E2**类型的度量来通告外部目的网络，那么路由器**C**将被选为最近的**ASBR**

3. 路由表的查找

当一台OSPF路由器检查一个数据包的目的地址时，它将通过下面的步骤来选择最优的路由：[\[19\]](#)

（1）选择可以和目的地址最精确匹配的路由。例如，如果路由表中存在路由条目172.16.64.0/18、172.16.64.0/24和172.16.64.192/27，而目的地址是172.16.64.205，那么最后一个路由条目将被选中。最精确的匹配应该总是最长匹配——拥有最长的地址掩码的路由。路由条目可以是主机地址、子网地址、网络地址、超网地址，或者缺省地址。如果路由器没有发现匹配的条目，那么它将发送一个ICMP目的不可达的消息给那个数据包的源地址，并且把这个数据包丢弃。

（2）通过排除次优的路径类型来剪除（prune）可选择条目的集合。路径类型根据下面的次序排列优先级，①表示最高的优先级，而④表示最低的优先级：

①区域内路径；

②区域间路径；

③E1外部路径；

④E2外部路径。

如果在最后的路由子集中还有多条等代价、等价路径类型的路由存在，那么OSPF协议将会利用它们。缺省条件下，Cisco路由器可以在最多16条等代价的路径上实现负载均衡（老的IOS版本中是4条），这个数值可以通过命令**maximum-paths** 来改变，改变的范围是1~6。 [\[20\]](#)

[8.1.5 认证](#)

OSPF协议对邻居路由器之间交换的所有数据包都具有认证的能力。认证可以是简单的口令认证或MD5加密校验和认证。这些认证的方法在第6章中已经讲述过了，本章将在后面的配置一节中给出一个配置OSPF认证的例子。

[8.1.6 按需电路上的OSPF](#)

OSPF协议每隔10s发送一次Hello数据包，并且每隔30min重新刷新一次它的LSA。这些功能用来维护邻接关系，以便确保链路状态数据库的精确，而且它比像RIP这样传统的距离矢量协议使用的带宽要少得多。然而，即使是一个很小的通信量，也不希望在按需电路上存在——例如像在X.25 SVC、ISDN和拨号电路等即用即连（usage-sensitive connection）的电路。这些链路可能根据连接次数或通信量或两者皆有来循环计算费用，因而，网络管理员必须减少它们的上线时间。

一个在按需电路上实用的OSPF协议的增强特性是，使OSPF具有抑制Hello数据包和LSA重刷新的能力，以便链路不需要永久的有效。 [\[21\]](#) 虽然这个增强特性是为即用即连的链路设计的，但是它在任何带宽有限的链路上也可能是有用的。 [\[22\]](#)

按需电路上的OSPF将会激活一条按需链路去执行最初的数据库同步，随后只会激活这条链路去泛洪扩散产生某些变化的LSA。这些变化是：

- LSA的可选字段发生了变化；
- 在老化时间达到MaxAge时收到了一个已经存在的LSA通告的新实例（instance）；

- LSA头部的长度字段发生了变化；
- LSA的内容发生了变化，但不包括20个八位组字节的头部、校验和或者序列号。

由于没有周期性的Hello数据包可以交换（Hello数据包只有在链路激活时才能使用），

OSPF协议必须有一个可达性的假定。也就是说，OSPF协议必须假定在需要链路连接的时候那条按需电路是可用的和有效的。但是在一些情况下，链路可能并不能立即可用和有效。例如，一个拨号链路可能正在使用，一个BRI链路的两个B信道可能也在使用，或者X.25所允许使用的最大的SVC电路数也都在使用。在这些情形下，链路并不是因为链路失效而变得不可用，而是因此链路太忙而变得不可用，这也是这种链路的正常特点，可以称为链路过忙（oversubscribed）。

OSPF协议将不会把链路过忙的按需链路报告为失效，因而数据包转发到一条过忙的链路上将会被丢弃而不是放入缓冲队列排队。这样做是有道理的，因为OSPF没有办法预先知道那条繁忙的链路什么时候可以再次变得可用，一连串数据包转发到这个不可用的接口上就会导致它们的缓冲区溢出。

关于接口状态机和邻居状态机，以及泛洪扩散过程的处理必须有几处修改，以便支持OSPF协议运行在按需电路上（更详细的内容请参考RFC1793）。在LSA通告的数据包格式里，也有两个地方需要修改。

首先，如果LSA通告没有通过按需电路进行周期性的重复刷新，那么经过LSA的最大生存时间（MaxAge）后，在这条链路的另一端将不会有路由器宣称这条LSA无效。OSPF协议更改了LSA的Age字段，它将Age字段的更高一位指定为DoNotAge位，以便解决这种情况的发生。当一条LSA在按需电路上进行泛洪扩散时，传送的路由器将把DoNotAge设置为1。这样，当这条LSA要泛洪扩散到这条链路另一端的所有路由器时，Age字段通常会增加一个InfTransDelay指定的秒数。[\[23\]](#)但是，当一条LSA被安置到路由器的链路状态数据库中后，这条LSA将不再像其他LSA一样老化。

第二个修改来自于第一个修改。因为所有的路由器必须能够正确地识别这个更改的DoNotAge位，因此在所有的LSA中增加了一个新的标志，

称为Demand Circuit位（DC-bit）。通过在所有LSA发起的时候设置这个标志位，路由器就可以通知其他路由器它是能够支持按需电路上的OSPF协议的。

按需电路上的OSPF（即指定老化字段的高位为DoNotAge位）带来的一个另外的好处是，我们现在可以在一个稳定的网络拓扑中减少泛洪扩散。在某个接口上使用命令**ip ospf flood-reduction**，DoNotAge位就可以设置在这个接口通告出去的LSA中，因此这些LSA在它们发生变化前不会重新刷新。

8.1.7 OSPF的数据包格式

OSPF数据包是由多重封装构成的，解析一个OSPF数据包就像给洋葱剥皮一样。正如图8-28所示，这个“洋葱”的外面一层是IP包的头部。封装在IP头部内的是5种OSPF数据包类型中的一种。每一种数据包类型都是由一个OSPF数据包头开始的，这个OSPF数据包头对于所有的数据包类型都是相同的。紧跟OSPF数据包头之后的是OSPF数据包数据，并且根据数据包类型的不同会有所不同。每一种数据包类型都有许多的特有类型字段（**type-specific fields**），后跟更多的数据包数据。在Hello数据包中，这些数据包数据包含的是邻居路由器的列表。在链路状态请求数据包中包含的是一系列描述被请求的LSA的字段。在链路状态更新数据包中包含的是一个LSA的列表，如图8-28中所示的那样。这些LSA依次都含有它们自己的头部和特有类型数据字段。数据库描述和链路状态确认数据包将包含有一个LSA头部的列表。

这里要注意，在一个网络上，OSPF数据包只能在邻居节点之间进行信息交换，它们从来都不会转发到始发它们的节点所在的网络。

图8-29显示了一台协议分析仪捕获到的传送OSPF数据的数据包的IP头部，表明OSPF的协议号是89。从这里可以看出，当OSPF数据包多播发生时，它们的TTL设置为1。既然一个OSPF数据包永远不应该跃过最近的邻居路由器转发，那么设置TTL为1可以帮助OSPF数据包确保自己的转发不会超过一跳。有一些路由器通过设置优先位（**precedence bit**）来运行一些具有优先次序的数据包进程。例如，这里的优先位可能是加权公平队列（**Weighted Fair Queuing, WFQ**）和加权随机预先检测

（**Weighted Random Early Detection, WRED**）等。正如图8-29所显示的，OSPF将设置优先位为互连网络控制（**Internetwork Control**,

110b)，以便这些具有优先次序的数据包进程给OSPF数据包一个高的优先级。

本小一节将从数据包头部开始，详细介绍这5种类型的OSPF数据包。随后的章节将详细介绍LSA数据包类型。在Hello数据包、数据库描述数据包和所有的LSA数据包中都含有一个可选（option）字段，这个字段的格式在所有的实例中都是一样的，因此将单独用一小节来详细介绍它。

1. 数据包头部

所有OSPF数据包都是由一个24个八位组字节的头部开始的，如图8-30所示。

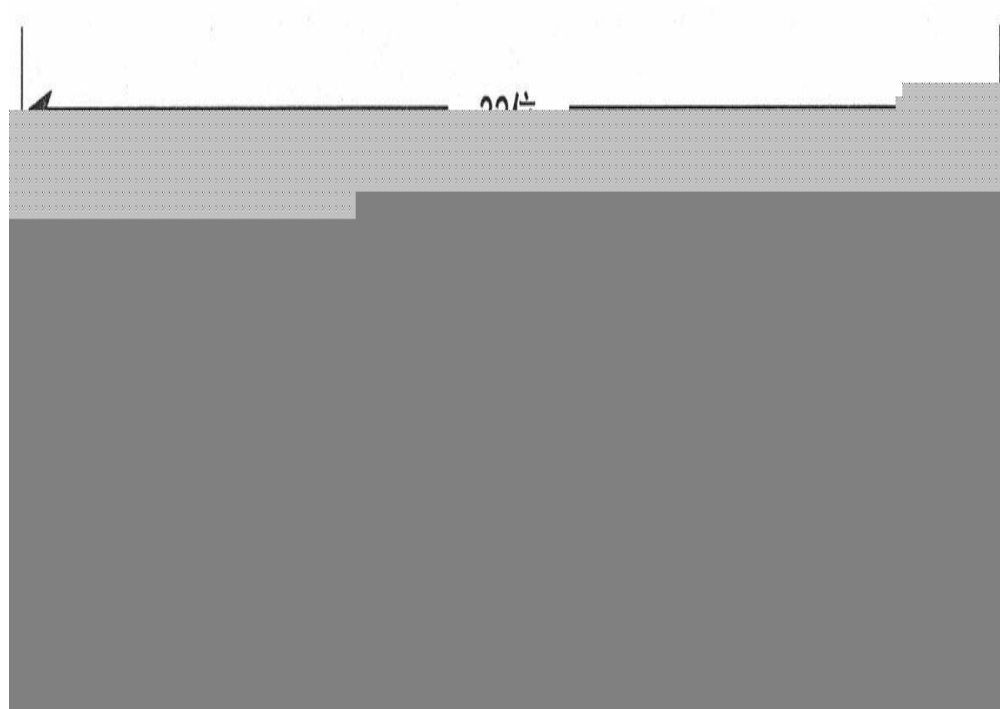


图8-28 一个OSPF数据包由一系列封装组成

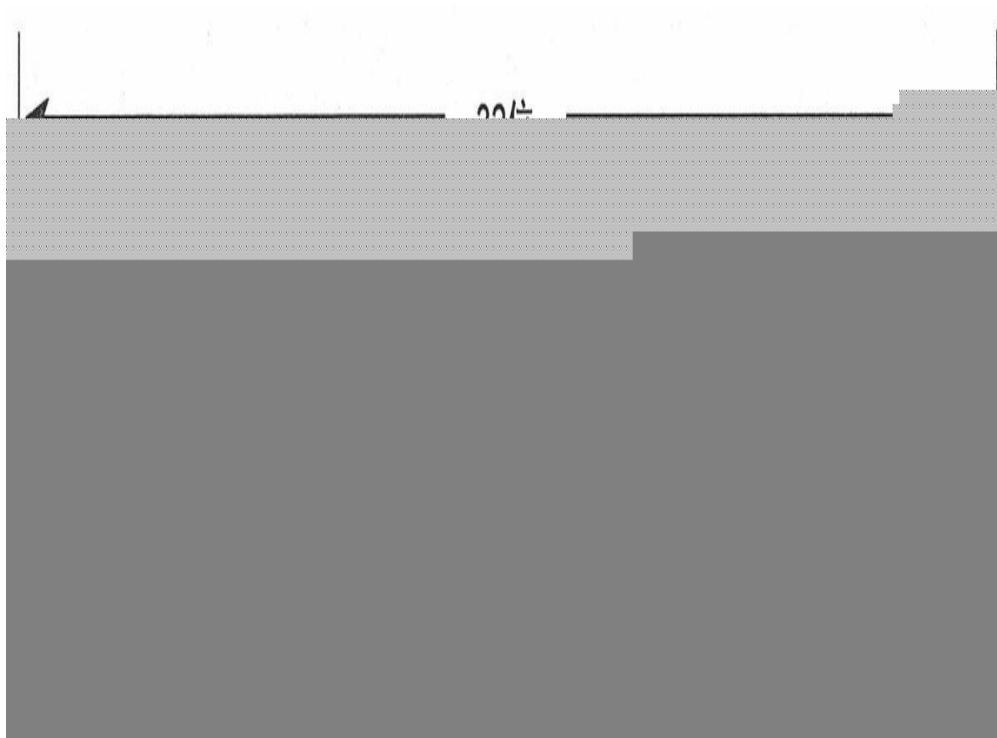


图8-29 OSPF使用的协议号是89。OSPF协议将IP头部的TTL值设置为1，并且把优先位设置成互连网络控制

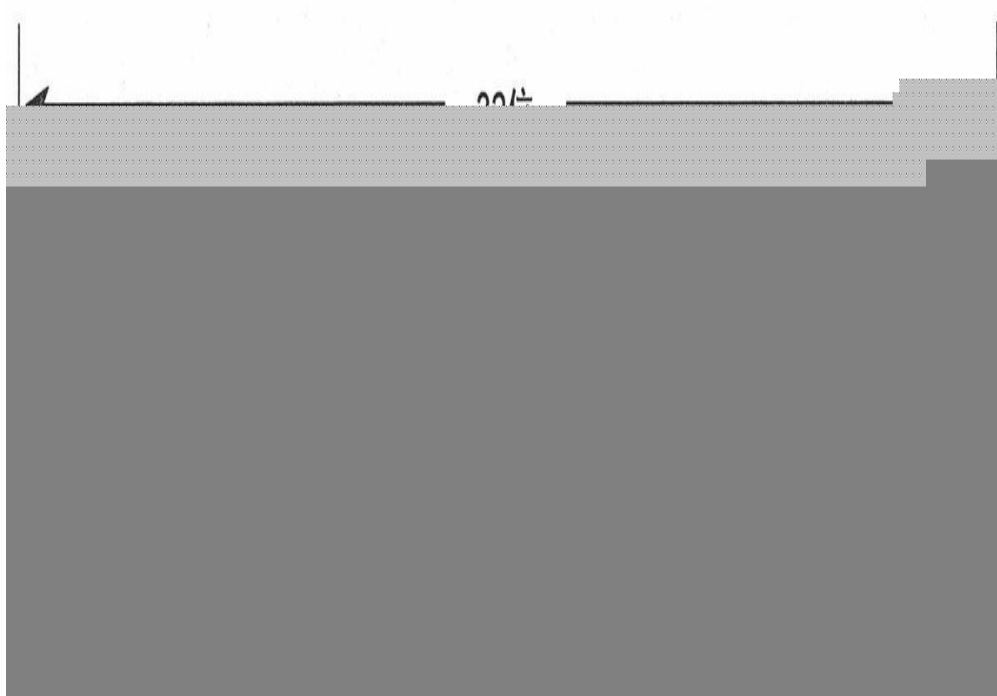


图8-30 OSPF包头

- 版本（**Version**）——是指OSPF的版本号。OSPF的版本号是2。对于IPv6的路由选择是OSPF版本3，OSPFv3将在下一章中讲述。
- 类型（**Type**）——指出跟在头部后面的数据包类型。根据出现在类型字段的数字，表8-7列出了这5种数据包类型。

表8-7 OSPF数据包类型

类型代码	描述
1	Hello
2	数据库描述
3	链路状态请求
4	链路状态更新
5	链路状态确认

- 数据包长度（**Packet Length**）——是指OSPF数据包的长度，包括数据包头部的长度，以八位组字节计。
- 路由器ID（**Router ID**）——是指始发路由器的ID。
- 区域ID（**Area ID**）——是指始发数据包的路由器所在的区域。如果数据包是在一个虚链路上发送的，那么区域ID就为0.0.0.0，也就是骨干区域的ID，因为虚链路被认为是骨干的一部分。
- 校验和（**Checksum**）——是指一个对整个数据包（包括包头）的标准IP校验和。
- 认证类型（**AuType**）——是指正在使用的认证模式。

表8-8中列出了可能的认证模式。

表8-8 OSPF认证类型

认证类型代码（ AuType ）	认证类型
0	空（没有认证）

1	简单（明文）口令认证
2	加密校验和（MD5）

- 认证（**Authentication**）——是指数据包认证的必要信息，认证可以是AuType字段中指定的任何一种认证模式。如果AuType=0，将不检查这个认证字段，因此可以包含任何内容。如果AuType=1，这个字段将包含一个最长为64位的口令。如果AuType=2，这个认证字段将包含一个Key ID、认证数据长度和一个不减小的加密序列号。这个消息摘要附加在OSPF数据包的尾部，不作为OSPF数据包本身的一部分。
- 密钥ID（**Key ID**）——标识认证算法和创建消息摘要使用的安全密钥。
- 认证数据长度（**Authentication Data Length**）——指明附加在OSPF数据包尾部的消息摘要的长度，以八位组字节计。
- 加密序列号（**Cryptographic Sequence Number**）——是一个不会减小的数字，用来防止重现攻击（replay attacks）。

2. Hello数据包

如图8-31所示，Hello数据包是用来建立和维护邻接关系的。为了形成一种邻接关系，Hello数据包携带的参数必须和它的邻居保持一致。

- 网络掩码（**Network Mask**）——是指发送数据包的接口的网络掩码。如果这个掩码和接收该数据包的接口的网络掩码不匹配，那么该数据包将被丢弃。这一技术性的措施可以确保路由器之间只有在它们共享网络的地址精确匹配时才能互相成为邻居。

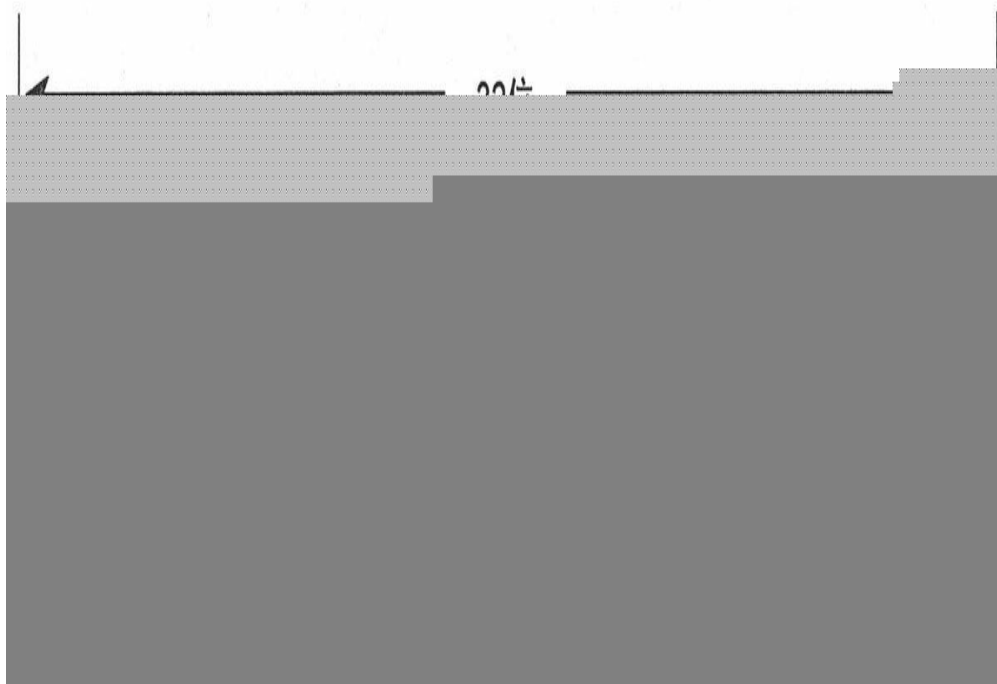


图8-31 OSPF协议Hello数据包

- **Hello**时间间隔（**Hello Interval**）——与前面讲述的一样，是指接口上Hello数据包传送之间的时间间隔，也是一个周期性的时间段，并以秒来计。对于这个参数，如果发送和接收路由器没有相同的值，那么它们就不能建立一种邻居关系。

- 可选项（**Option**）——这个字段将在8.1.9小节中讲述。Hello数据包中包含的这个字段可以用来确保邻居之间的兼容性问题。一台路由器可以拒绝一台兼容性不匹配的邻居路由器。

- 路由器优先级（**Router Priority**）——是用来做DR和BDR路由器的选举的。如果

该字段设置为0，那么始发路由器将没有资格被选成DR和BDR路由器。

- 路由器无效时间间隔（**Router Dead Interval**）——是指始发路由器在宣告邻居路由器无效之前，将要等待从邻居路由器发出的Hello数据包的时长，以秒数计。如果路由器收到Hello数据包的时间和接收接口配置的RouterDeadInterval不匹配，那么这个Hello数据包将被丢弃。这种做法可以确保邻居之间的这个参数的一致性。

- 指定路由器（**DR**）——是指网络上指定路由器接口的IP地址，注意，这里指的不是指定路由器的路由器ID。在选取DR的过程中，这可能只是始发路由器所认为的DR，而不是最终选举出来的DR。如果没有DR（因为DR可能还没有选出或者网络类型根本不需要DR），那么这个字段就会被设置为0.0.0.0。
- 备份指定路由器（**BDR**）——是指网络上备份指定路由器接口的IP地址。同样的，在选举BDR的过程中，这可能只是始发路由器所认为的BDR，而不是最终选举出来的BDR。如果没有BDR，那么这个字段就会被设置为0.0.0.0。
- 邻居（**Neighbor**）——是一个递归字段，如果始发路由器在过去的一个Router DeadInterval时间内，从网络上已经收到来自它的某些邻居的有效Hello数据包，那么将会在这个字段中列出所有这些邻居的RID。

3. 数据库描述数据包

如图8-32所示，数据库描述数据包用于正在建立的邻接关系（请参考本章前面“建立一个邻接关系”部分的内容）。数据库描述数据包的一个主要目的是描述始发路由器数据库中的一些或者全部LSA信息，以便接收路由器能够确定所收到的LSA在其数据库中是否已经有一个匹配的LSA。这个操作只需要列出LSA的头部就可以完成。LSA头部包括了足够多的信息，不仅可以标识一个特定的LSA，还可以标识该LSA的最近实例。由于在数据库描述（DD）处理过程中，可能需要交换多个数据库描述数据包，因此数据库描述数据包中包含了一个主/从控制关系的标志，用来管理这些数据包的交换。

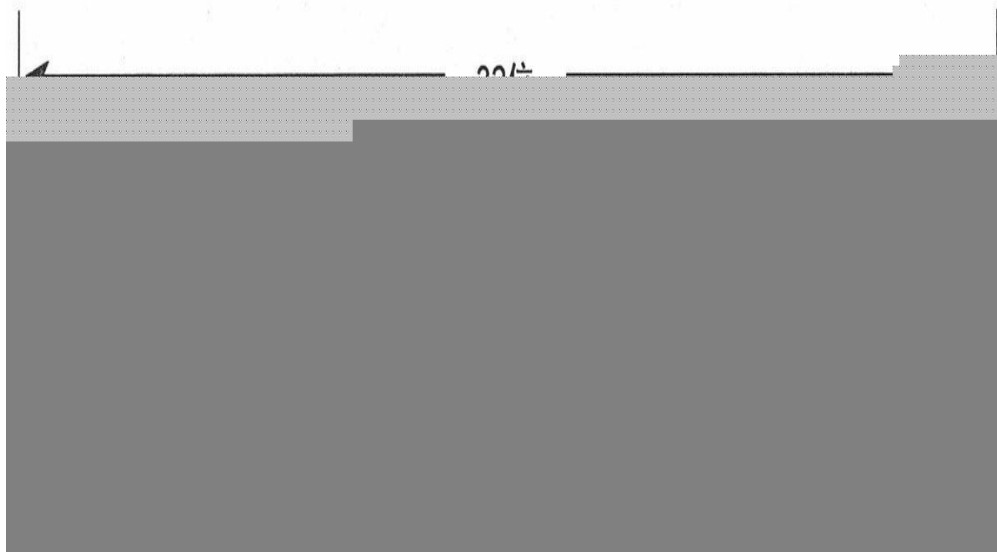


图8-32 OSPF数据库描述数据包

- **接口MTU（Interface MTU）** ——是指在数据包不分段的情况下，始发路由器接口可以发送的最大IP数据包大小，以八位组字节计。当数据包在虚链路上传送时，这个字段的值设置为0x0000。
- **可选项（Option）** ——这个字段将在8.1.9小节中讲述。该字段包含在数据库描述数据包中，使路由器可以选择转发某些LSA到那些没有必要的支持能力的邻居路由器。

报文下一个八位组字节的前5位没有被使用而总是设置为00000b。

- **I位，或称为初始位（Initial bit）** ——当发送的是一系列数据库描述数据包中的最初一个数据包时，该位设置为1。后续的数据包描述数据包将把该位设置为0，即I-bit=0。
- **M位，或称为后继位（More bit）** ——当发送的数据包还不是一系列数据库描述数据包中的最后一个数据包时，将该位设置为1。最后的一个数据库描述数据包将把该位设置为0，即M-bit=0。
- **MS位，或称为主/从位（Master/Slave bit）** ——在数据库同步过程中，该位设置为1，用来指明始发数据库描述数据包的路由器是一台“主”路由器（也就是说，是主从关系协商过程的控制者）。“从”路由器将该位设置为0，即MS-bit=0。

- **数据库描述序列号（DD Sequence Number）** ——在数据库的同步过程中，用来确保路由器能够收到完整的数据库描述数据包序列。这个序列号将由“主”路由器在最初发送的数据库描述数据包中设置一些惟一的数值，而后续数据包的序列号将依次增加。
- **LSA头部（LSA Header）** ——列出了始发路由器的链路状态数据库中部分或全部LSA头部。参见“链路状态头部”，那里有一个关于LSA头部的完整描述。在LSA头部里包含有足够的信息可以惟一地标识一个LSA和一个LSA的具体实例。

4. 链路状态请求数据包

在数据库同步过程中如果收到了数据库描述数据包，路由器将会查看数据库描述数据包里有哪些LSA不在自己的数据库中，或者有哪些LSA比自己数据库中的LSA更新。然后，将把这些LSA记录在链路状态请求列表中。接着，路由器会发送一个或多个链路状态请求数据包去向它的邻居请求发送在链路状态请求列表中的这些LSA的副本，如图8-33所示。这里要注意，一个数据包可以根据一个LSA头部的类型、ID和通告路由器进行惟一的标识，但是它不能请求这个LSA的具体实例（LSA的具体实例由LSA头部的序列号、校验和以及老化时间标识）。因此，不论请求路由器是否知道是LSA的哪个具体实例，它所请求的都是LSA的最新实例。这个过程避免了这样一种情形——在路由器最新描述LSA到其副本被请求的时间之间，邻居可能获得或发起一个更新的LSA副本。

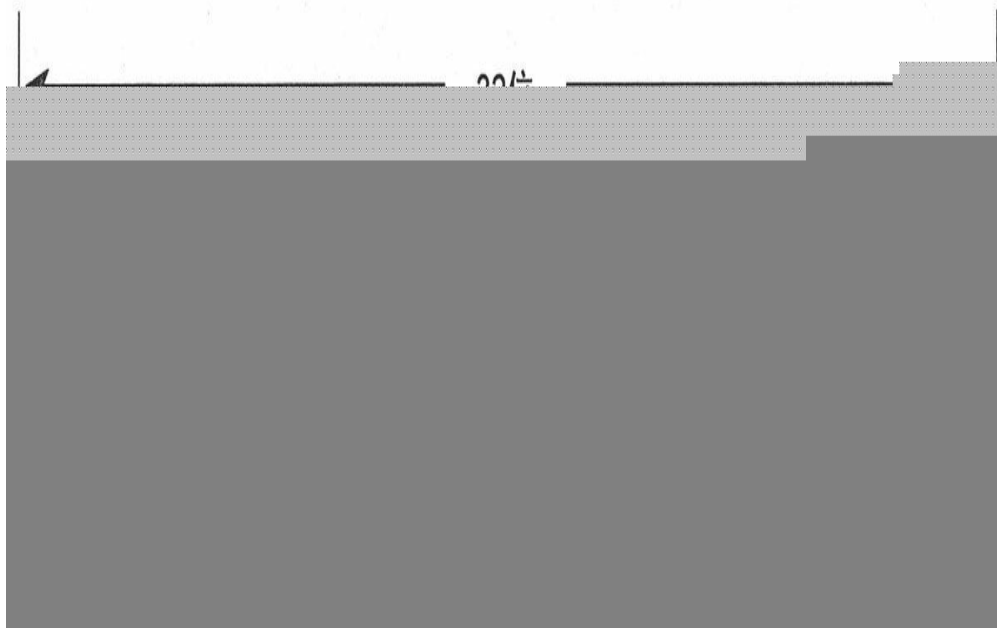


图8-33 OSPF链路状态请求数据包

- 链路状态类型（**Link State Type**）——是一个链路状态类型号，用来指明LSA标识是一个路由器LSA、一个网络LSA还是其他类型的LSA，等等。表8-4中列出了这些类型号。
- 链路状态ID（**Link State ID**）——是LSA头部中和类型无关的字段。请参见“链路状态头部”和具体介绍LSA的章节可以得到关于不同类型的LSA如何使用该字段的完整描述。
- 通告路由器（**Advertising Router**）——是指始发LSA通告的路由器的路由器ID。

5. 链路状态更新数据包

如图8-34所示，链路状态（LS）更新数据包是用于LSA的泛洪扩散和发送LSA去响应链路状态请求数据包的。请记住，OSPF数据包是不能离开发起它们的网络的。因此，一个链路状态数据包可以携带一个或多个LSA，但是这些LSA只能传送到始发它们的路由器的直连邻居。接收LSA的邻居路由器将负责在新的LS更新数据包中重新封装相关的LSA，从而进一步泛洪扩散到它自己的邻居。

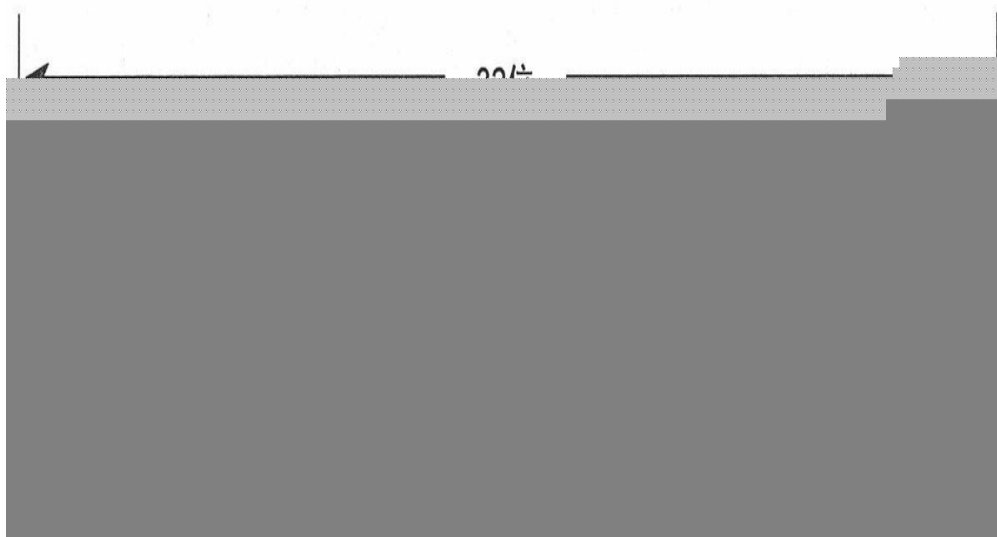


图8-34 OSPF链路状态更新数据包

- **LSA数量（Number of LSA）** ——指出这个数据包中包含的LSA的数量。
- **链路状态通告（LSA）** ——是指在OSPF协议的LSA数据包格式中描述的全部LSA。每一个更新数据包都可以携带多个LSA，它的大小可以达到传送该数据包的链路所允许的最大数据包尺寸。

6. 链路状态确认数据包

链路状态确认数据包是用来进行LSA可靠的泛洪扩散的。一台路由器从它的邻居路由器收到的每一个LSA都必须在链路状态确认数据包中进行明确的确认。被确认的LSA是根据在链路状态确认数据包里包含它的头部来辨别的，并且多个LSA可以通过单个数据包来确认。正如图8-35所显示的，一个链路状态确认数据包的组成除了OSPF包头和一个LSA头部的列表之外，就没有其他多余的内容了。

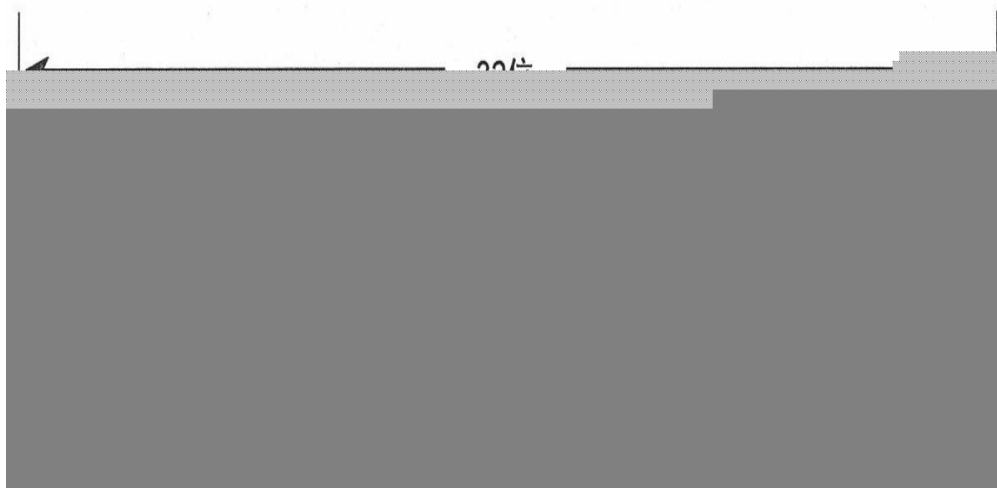


图8-35 OSPF链路状态确认数据包

8.1.8 OSPF的LSA格式

本小节将详细描述类型1~类型5和类型7的LSA字段的含义，不讨论组成员LSA（类型6）是因为MOSPF协议不在本书的讲述范围，同样地，类型8~类型11的LSA也不讨论，因为它们或者还没有部署（类型8），或者它们支持的一些特性已经超出了本书的范围。

1. LSA的头部

LSA头部在所有LSA的开始处，如图836所示。在数据库描述数据包和链路状态确认数据包里也使用了LSA的头部本身。在LSA头部中有3个字段可以惟一地识别每个LSA：类型、链路状态ID和通告路由器。另外，还有其他3个字段可以惟一地识别一个LSA的最新实例：老化时间、序列号和校验和。



图8-36 OSPF协议LSA头部

- 老化时间（**Age**）——是指自从发出LSA后所经历的时间，以秒数计。当泛洪扩散LSA时，在从每一台路由器接口转发出去时，LSA的老化时间都会增加一个InfTransDelay的秒数。当然，当LSA驻留在链路状态数据库内时，这个老化时间也会增大。
- 可选项（**Option**）——这个字段将在8.1.9小节中讲述。在LSA的头部中，该字段指出了在部分OSPF域中LSA能够支持的可选性能。
- 类型（**Type**）——就是LSA的类型。一些类型的代码可以参见表8-4。
- 链路状态ID（**Link State ID**）——用来指定LSA所描述的部分OSPF域。这个字段的特殊用法根据LSA的类型而会有所不同。每一个LSA的描述都包含了一个怎样使用这个字段的描述。
- 通告路由器（**Advertising Router**）——是指始发LSA的路路由器的ID。
- 序列号（**Sequence Number**）——当LSA每次有新的实例产生时，这个序列号就会增加。这个更新可以帮助其他路由器识别最新的LSA实例。
- 校验和（**Checksum**）——这是一个除了Age字段之外，关于LSA的全部信息的校验和。因为如果包含了Age字段，那么这个校验和将会随着老化时间的增大而每次都需要进行重新计算。
- 长度（**Length**）——是一个包含LSA头部在内的LSA的长度，用八位组字节表示。

2. 路由器LSA

如图8-37所示，路由器LSA是由每一台路由器产生的。它列出了一台路由器的链路或接口，同时也列出了这些接口的状态和每一条链路的出站代价。这些路由器LSA只能在始发它们的OSPF区域内进行泛洪扩散。使用命令**show ip ospf database router** 可以列出链路状态数据库里的路

由器LSA（请参见示例8-11）。这里要注意，路由器LSA是把主机路由作为末梢网络来通告的，它的链路ID字段携带的是主机的IP地址，而链路数据字段携带的是主机地址的掩码——255.255.255.255。

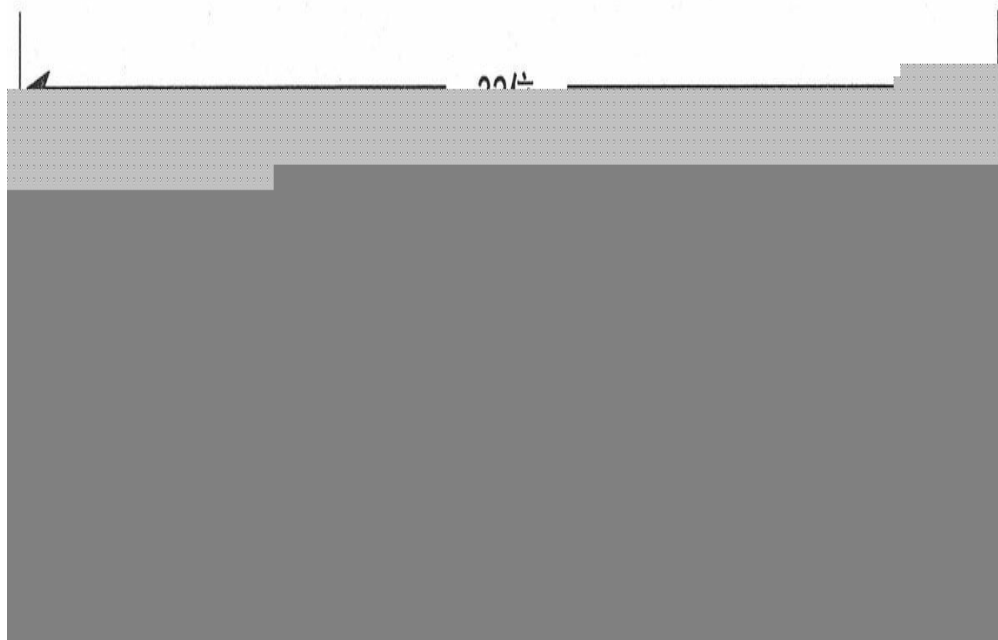


图8-37 OSPF的路由器LSA

- **链路状态ID（Link State ID）** ——路由器LSA的链路状态ID是指始发路由器的路由器ID。
- **V，或虚链路端点位（Virtual Link Endpoint bit）** ——设置为1时，说明始发路由器是一条或多条具有完全邻接关系的虚链路的一个端点，这里被描述的区域是传送区域。
- **E，或外部位（External bit）** ——当始发路由器是一个ASBR路由器时，设置该位为1。
- **B，或边界位（Border bit）** ——当始发路由器是一个ABR路由器时，设置该位为1。
- **链路数量（Number of Links）** ——标明一个LSA所描述的路由器链路数量。对于LSA进行泛洪扩散的区域，路由器LSA必须描述始发路由器的所有链路或接口。

在路由器LSA后续的字段里描述了每一条链路，并且出现的次数和前面链路数量字段中的数量是一致的。虽然这个字段是出现在链路数据字段之后的，但是在这里将首先讲述链路类型字段。这是因为链路ID和链路数据字段的描述会根据链路类型字段的值而有所变化，因此首先理解链路类型是必要的。

- **链路类型（Link Type）**——描述了链路所提供连接的一般类型。表8-9列出了这个字段可能的值和相关的连接类型。

表8-9 链路类型的值

链路类型	连接
1	点到点连接到另一台路由器
2	连接到一个传送网络
3	连接到一个末梢网络
4	虚链路

- **链路ID（Link ID）**——用来标识链路连接的对象。这个字段依赖于表8-10中的链路类型字段。注意，当连接的对象是另一台路由器时，链路ID和在邻居路由器的LSA头部的链路状态ID是相同的。在计算路由表的期间，这个值可以用来发现链路状态数据库中邻居的LSA。

表8-10 链路ID的值

链路类型	链路ID字段的值
1	邻居路由器的路由器ID
2	DR路由器的接口的IP地址
3	IP网络或子网地址
4	邻居路由器的路由器ID

- 链路数据（**Link Data**）——也是依赖于链路类型字段值的字段，如表8-11所示。

表8-11 链路数据的值

链路类型	链路数据字段的值
普通链路	0
聚合链路	1
虚链路	2
物理链路	3

- | | |
|---|-------------------------|
| 1 | 和网络相连的始发路由器接口的IP地址* |
| 2 | 和网络相连的始发路由器接口的IP地址 |
| 3 | 网络的IP地址或子网掩码 |
| 4 | 始发路由器接口的MIB-II ifIndex值 |

* 如果点到点链路是无编号的，那么这个字段将替代携带接口的MIB-II ifIndex值。

- **ToS号（Number of ToS）** ——为列出的这条链路指定服务类型度量的编号。虽然RFC 2328已经不再支持ToS，但是为了向前兼容早期部署的OSPF，仍旧保留这个字段。如果没有ToS度量和一条链路相关联，那么这个字段就设置为0x00。
- **度量（Metric）** ——是指一条链路（接口）的代价。

接下来的两个与链路相关联的字段是和ToS号（#）字段一致的。如果ToS的#=3，那么将有3个32位字包含这些字段的3个实例。如果ToS的#=0，那么将没有这些字段的实例。

注意，Cisco路由器只支持ToS=0。

- **ToS** ——指定了后面提及的度量涉及到的服务类型。[\[24\]](#)表8-12列出了ToS的值（在RFC 1349中指定的）、在IP头部中相应的比特值和在OSPF ToS字段中使用的相应值。

表8-12 OSPF ToS的值

RFC ToS的值	IP头部ToS字段	OSPF的ToS编码
正常的服务	0000	0
最小的成本代价	0001	2
最大的可靠性	0010	4
最大的吞吐量	0100	8
最小的时延	1000	16

- **ToS度量（ToS Metric）** ——和指定的ToS值相关联的度量。

3. 网络LSA

如图8-38所示，网络LSA是始发于指定路由器（DR）的。这些网络LSA将通告一个多路访问网络和与这个网络相连的所有路由器（包括DR）。像路由器LSA一样，网络LSA也只能在始发这条网络LSA的区域内进行泛洪扩散。可以使用**show ip ospf database network** 来查看一条网络LSA，参见示例8-12。



图8-38 OSPF的网络LSA

- 链路状态ID（**Link State ID**）——网络LSA的链路状态ID是指网络中DR路由器接口上的IP地址。
- 网络掩码（**Network Mask**）——指定这个网络上使用的地址或子网的掩码。
- 相连的路由器（**Attached Router**）——列出了多路访问网络上所有与DR形成完全邻接关系的路由器的路由器ID，以及DR路由器本身的路由器ID。这个字段的实例数量（也是列出的路由器的数量）可以由LSA头部的长度字段推断出来。

4. 网络汇总LSA和ASBR汇总LSA

网络汇总LSA（类型3）和ASBR汇总LSA（类型4）具有同样的格式，如图8-39所示。在它们的字段内容里，惟一的不同之处是它们所指的类

型和链路状态ID。ABR路由器将产生这两种类型的汇总LSA。网络汇总LSA通告的是一个区域外部的网络（包括缺省路由），而ASBR汇总LSA通告的是一个区域外部的ASBR路由器。这两种类型的LSA都只能泛洪扩散到单个区域。使用命令**show ip ospf database summary** 可以查看一台路由器的链路状态数据库中的网络汇总LSA，参见示例8-13所示。而使用命令**show ip ospf database asbr-summary** 可以查看链路状态数据库中的ASBR汇总LSA，参见示例8-14。

- **链路状态ID（Link State ID）** ——对于类型3的LSA来说，它是所通告的网络或子网的IP地址。对于类型4的LSA来说，链路状态ID是所通告的ASBR路由器的路由器ID。
- **网络掩码（Network Mask）** ——在类型3的LSA中，是指所通告的网络的子网掩码或地址。在类型4的LSA中，这个字段没有什么实际意义，并被设置为0.0.0.0。

如果一条类型3的LSA通告的是一条缺省路由，那么链路状态ID和网络掩码字段都将是0.0.0.0。

- **度量（Metric）** ——是指到达目的地的路由的代价。

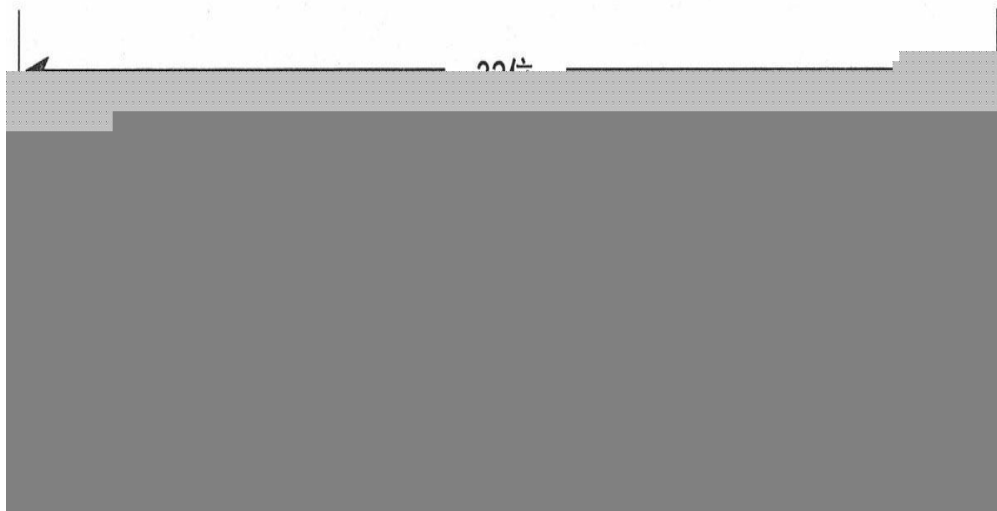


图8-39 OSPF的汇总LSA。类型3和类型4的汇总LSA具有同样的格式

ToS字段和ToS度量字段都是可选字段，并且已经在“路由器LSA”部分描述过了。另外提醒一下，Cisco路由器只支持ToS=0。

5. 自主系统外部LSA

如图8-40所示，自主系统外部LSA是由ASBR路由器始发的。这些自主系统外部LSA是用来通告OSPF自主系统外部的目的网络的，这里也包括到达外部目的网络的缺省路由。自主系统外部LSA可以泛洪扩散到OSPF域中所有非末梢区域中去。可以使用命令**show ip ospf database external** 来查看AS外部LSA，参见示例8-15。

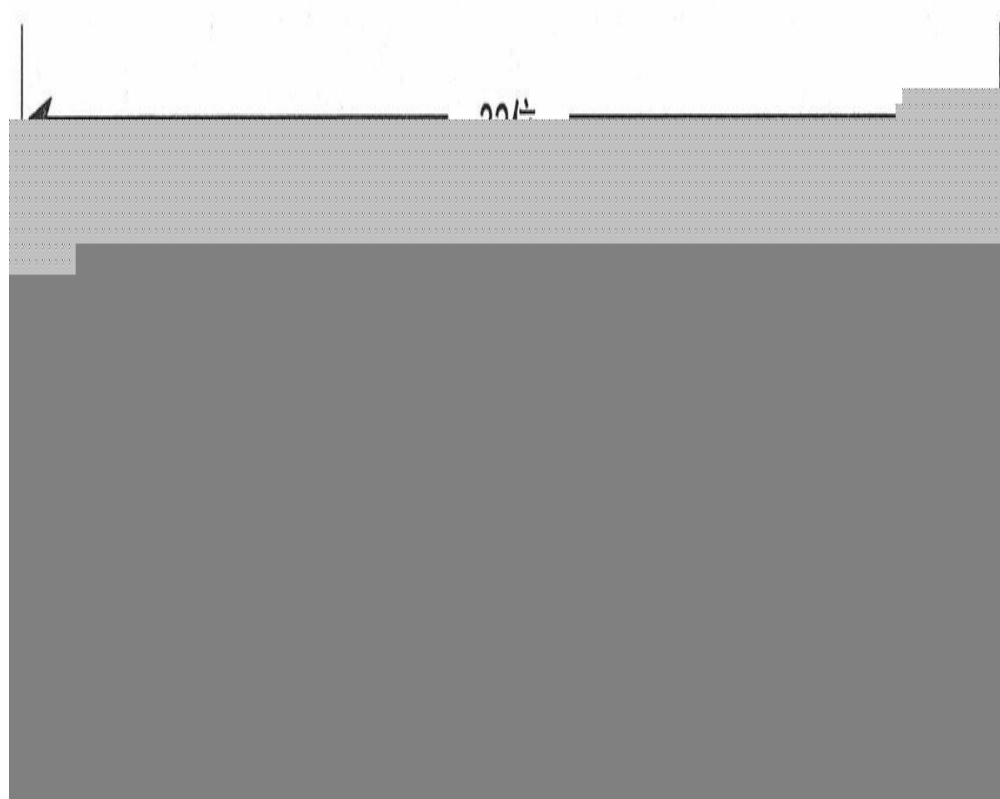


图8-40 OSPF的自主系统外部LSA

- **链路状态ID** ——自主系统外部LSA的链路状态ID是指目的地的IP地址。
- **网络掩码** ——是指所通告的目的地的子网掩码或地址。

如果类型5的LSA正在通告的是一条缺省路由，那么链路状态ID和网络掩码字段都将被设置为0.0.0.0。

- **E**，或称外部度量位（**External Metric bit**） ——用来指定这条

路由使用的外部度量的类型。如果该E-bit设置为1，那么度量类型就是E2；如果该E-bit设置为0，那么度量类型就是E1。请参考本章前面讲述的“路径类型”部分的内容，那里可以找到关于E1和E2外部度量类型的更多信息。

- 度量 ——是指路由的代价，由ASBR路由器设定。
- 转发地址（**Forwarding Address**） ——是指到达所通告的目的地的数据包应该被转发到的地址。如果转发地址是0.0.0.0，那么数据包将被转发到始发ASBR上。
- 外部路由标志（**External Route Tag**） ——是一个应用于外部路由的任意标志。OSPF协议本身并不使用这个字段，而是由外部路由来管理和控制。该标志的设定和用法将在第14章中介绍。

可选地，ToS字段也可以和某个目的地相关联。这些字段和前面讲述的是相同的，只是每一个ToS度量也都有自己的E-bit、转发地址和外部路由标志。

6. NSSA外部LSA

NSSA外部LSA是由一个NSSA区域内的ASBR路由器始发的。如图8-41所示，除了转发地址字段外，NSSA外部LSA的所有字段都和一个AS外部LSA的字段相同。不像AS外部LSA那样是在整个OSPF自主系统中进行泛洪扩散的，NSSA外部LSA仅仅在始发它们的一个非纯末梢区域中进行泛洪扩散。使用命令**show ip ospf database nssa-external** 可以显示出NSSA外部LSA的信息，参见示例8-16所示。

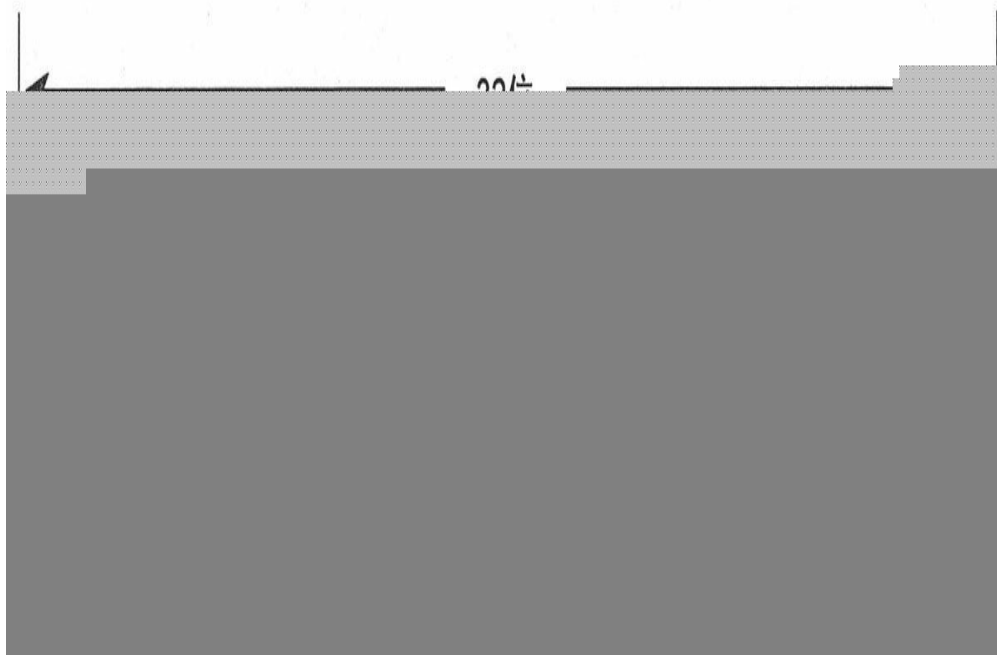


图8-41 OSPF的NSSA外部LSA

- 转发地址 ——如果网络在一台NSSA ASBR路由器和邻接的自主系统之间是作为一条内部路由通告的，那么这个转发地址就是指这个网络的下一跳地址。如果网络不是作为一条内部路由通告，那么这个转发地址将是NSSA ASBR路由器的路由器ID。

8.1.9 可选项字段

如图8-42所示，可选字段是出现在每一个Hello数据包、数据库描述数据包和每一个LSA中的。可选项字段允许路由器和其他路由器进行一些可选性能的通信。



图8-42 OSPF的可选项字段

- **DN** ——用于基于MPLS的第3层虚拟专用网（VPN），在RFC规定VPN之后通常称为RFC 2547 VPN。当一条路由通过OSPF从某个客户网络学到后，它就会穿过使用多协议BGP（Multiprotocol

BGP)的RFC 2547 VPN被通告到网络对端，接着再通过OSPF被通告回客户网络。通告回的OSPF路由在BGP中会被重新分配到VPN运营商的网络，这样就会产生一个环路。DN位用来避免这个环路。当在类型3、类型5或类型7的LSA中设置了DN位后，接收路由器就不能在它的OSPF路由计算中使用该LSA。

- **O** ——设置用来表明始发路由器支持Opaque LSA（类型9，类型10和类型11）。
- **DC**位 ——当始发路由器具有支持按需电路上的OSPF的能力时，该位将被设置。
- **EA**位 ——当始发路由器具有接收和转发外部属性LSA的能力时，该位将被设置。这些LSA还没有一般的用法，因此本书将不介绍它们。
- **N**位 ——只用在Hello数据包中。一台路由器设置N-bit=1表明它支持NSSA外部LSA。如果设置N-bit=0，那么路由器将不接受和发送NSSA外部LSA。邻居路由器如果错误配置了N-bit将不会形成邻接关系，这个限制可以确保一个区域内的所有路由器都同样地具有支持NSSA的能力。如果N-bit=1，那么E-bit必须设置为0。
- **P**位 ——只用在NSSA外部LSA的头部（由于这种情况，N-bit和P-bit可以使用在同一位置）。该位将告诉一个非纯末梢区域中的ABR路由器将类型7的LSA转换为类型5的LSA。
- **MC**位 ——当始发路由器具有转发IP组播数据包的能力时，该位将被设置。这一位使用在MOSPF协议当中。
- **E**位 ——当始发路由器具有接受AS外部LSA的能力时，该位将被设置。在所有的AS外部LSA和所有始发于骨干区域以及非末梢区域的LSA中，该位将设置为1。而在所有始发于末梢区域的LSA当中，该位设置为0。另外，可以在Hello数据包中使用该位来表明一个接口具有接收和发送类型5的LSA的能力。E-bit配置错误的邻居路由器将不能形成邻接关系，这个限制可以确保一个区域的所有路由器都同样地具有支持末梢区域的能力。
- **MT**位 ——设置了该位表示始发路由器支持多拓扑OSPF（MT-

OSPF)。在本书编写期间，MT-OSPF还仅仅是一个提议，并未被广泛采用。

旧的OSPF标准规定，目前被MT位占用的可选位是T位。设置了T位表示始发路由器具有支持ToS的能力。但是，由于ToS特性从来没有部署过，所以T位也就从来没有使用过。

8.2 配置OSPF

在一个大型的IP网络中，有很多可用的OSPF选项和配置变量经常用于IGP。然而，偶尔会听到这样一种观点，认为在一个小型的网络中使用OSPF协议不是一种好的选择，因为OSPF协议的配置显得“太复杂”了。这纯粹是无稽之谈。正如下面所介绍的第一个配置案例中显示的，完成一个基本的OSPF配置并使OSPF协议可以正常地运行，只需要在network命令中额外地敲几下键盘而已。如果对OSPF的操作已经有了相当的理解，那么所输入的这些额外的内容也显得十分直观和自然了。

8.2.1 案例研究：一个基本的OSPF配置

配置一个基本的OSPF的过程含有以下3个必要的步骤：

步骤1： 确定和每一个路由器接口相连的区域。

步骤2： 使用**router ospf process-id** 命令来启动一个OSPF进程。

步骤3： 使用**network area**命令来指定运行OSPF协议的接口和它们所在的区域。

与IGRP和EIGRP协议中相关的进程ID不同，OSPF协议的进程ID不是一个自主系统号。OSPF的这个进程ID可以是任何正整数，并且仅在配置它的路由器内有意义。Cisco IOS软件允许同一台路由器中运行多个OSPF进程，[\[25\]](#)进程ID不过是在同一台设备中用来区分一个进程与另一个不同的进程而已。

在RIP协议中，命令**network** 只允许用来指定一个主网络地址。如果在这个网络中的一些接口不能运行该路由选择协议的话，那么就在这些协议中使用**passive-interface** 命令来抑制这些接口。正如在EIGRP协议中使用带通配符掩码的**network** 命令，使用命令**network area** 比RIP与EIGRP在引入掩码选项之前使用的**network** 命令更加灵活，它可以完全反映OSPF协议的无类别特性。任何一个地址范围都能够使用一个（地址，反向掩码）对来指定。这里的反向掩码（**inverse mask**）和在访问列表中使用反向掩码是一样的。[\[26\]](#)对于区域的指定，既可以使用一个十进制数字表示，也可以使用一个点分十进制数来表示。

图8-43显示了一个OSPF的网络。注意，在这里每一个区域都具有一个指定的IP地址，这个地址源于它的子网。限制一个区域为一个地址或子网是不必要的，但是这样做会带来一些有效的好处，这在后面的8.2.8小节中将会看到。注意，这个例子是设计用来演示多个区域的配置的。在现实的网络环境中，将图中这样的一个小网络设计成单个区域应该是一种更聪明的做法。更进一步来说，这里的单个区域也不必一定是区域0。OSPF的规则规定所有的区域必须连接到骨干区域；因此，只有当有多于一个的区域时才需要骨干区域。

在图8-43中的4台路由器上，每一台路由器的配置都不同，这样做是为了更好地说明**network area** 命令的灵活性。相关的配置参见示例8-19～示例8-22。

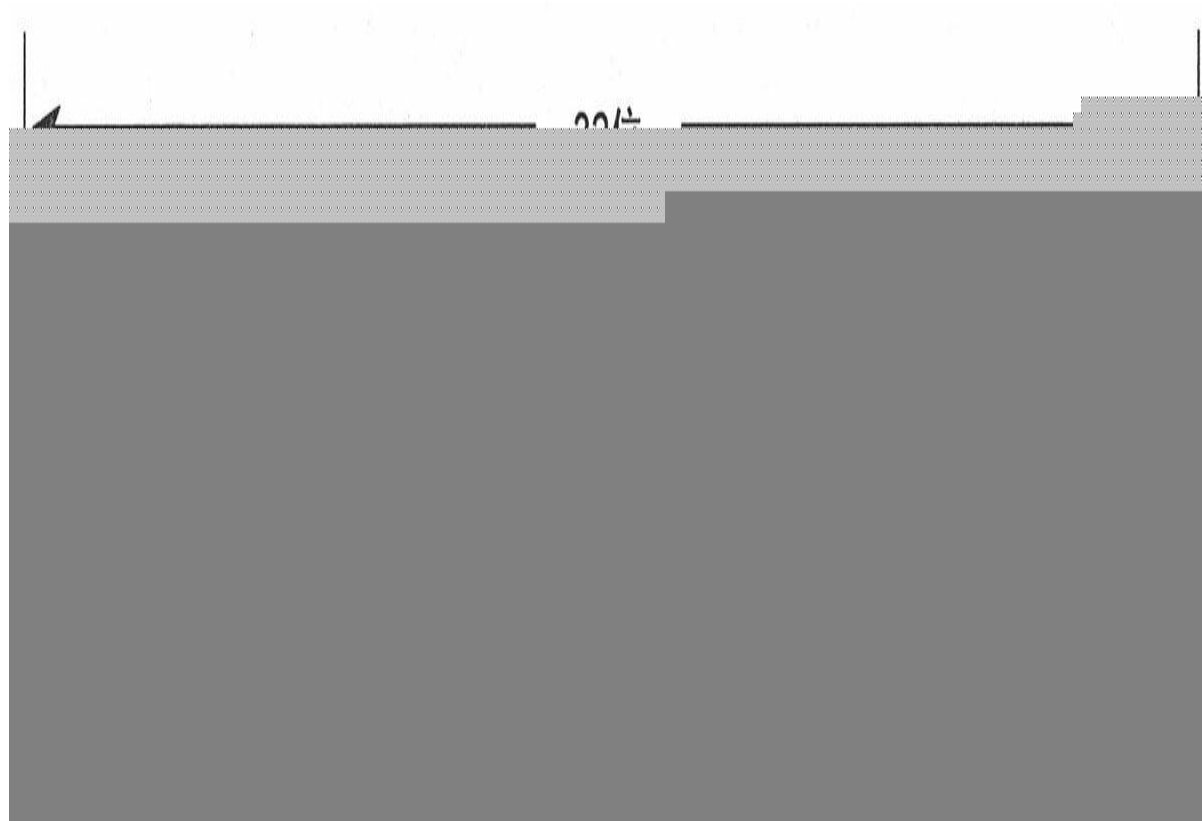


图8-43 路由器Chardin和Goya都是ABR路由器，而路由器Rubens和Matisse是内部路由器

示例8-19 路由器Rubens的OSPF网络区域配置

示例8-20 路由器Chardin的OSPF网络区域配置

示例8-21 路由器Goya的OSPF网络区域配置

示例8-22 路由器Matisse的OSPF网络区域配置

这里要注意的第一件事情是，在每一台路由器上配置的进程ID都是不同的。一般情况下，为了保持在一个网络中配置的一致性，这些数字是相同的。而在这里，配置不同的进程ID号只不过是为了演示这样一个事实——它们在本地图器之外是没有意义的。当然，这4个不同编号的进程是可以相互通信的。

要注意的第二件事情就是**network area** 命令的格式。紧跟在**network** 之后的部分是一个IP地址和一个反向掩码。当OSPF进程第一次启动时，它将根据第一个网络语句（即第一个**network** 语句）的（地址，反向掩码）对来“启动”所有有效接口的IP地址。所有匹配的接口将被分配到根据**network** 命令的**area** 部分指定的区域。接着，这个过程根据第二条**network** 语句继续匹配启动所有不匹配第一条**network** 语句的接口。这样，根据一条条**network** 语句 运行IP地址的过程一直持续到所有的接口都匹配，或者所有的**network** 语句都被使用为止。这里要注意有一点非常重要，就是这个过程从第一条**network** 语句开始是有顺序处理的。正如在后面故障排除一节中所描述的，**network** 语句的顺序变得很重要。

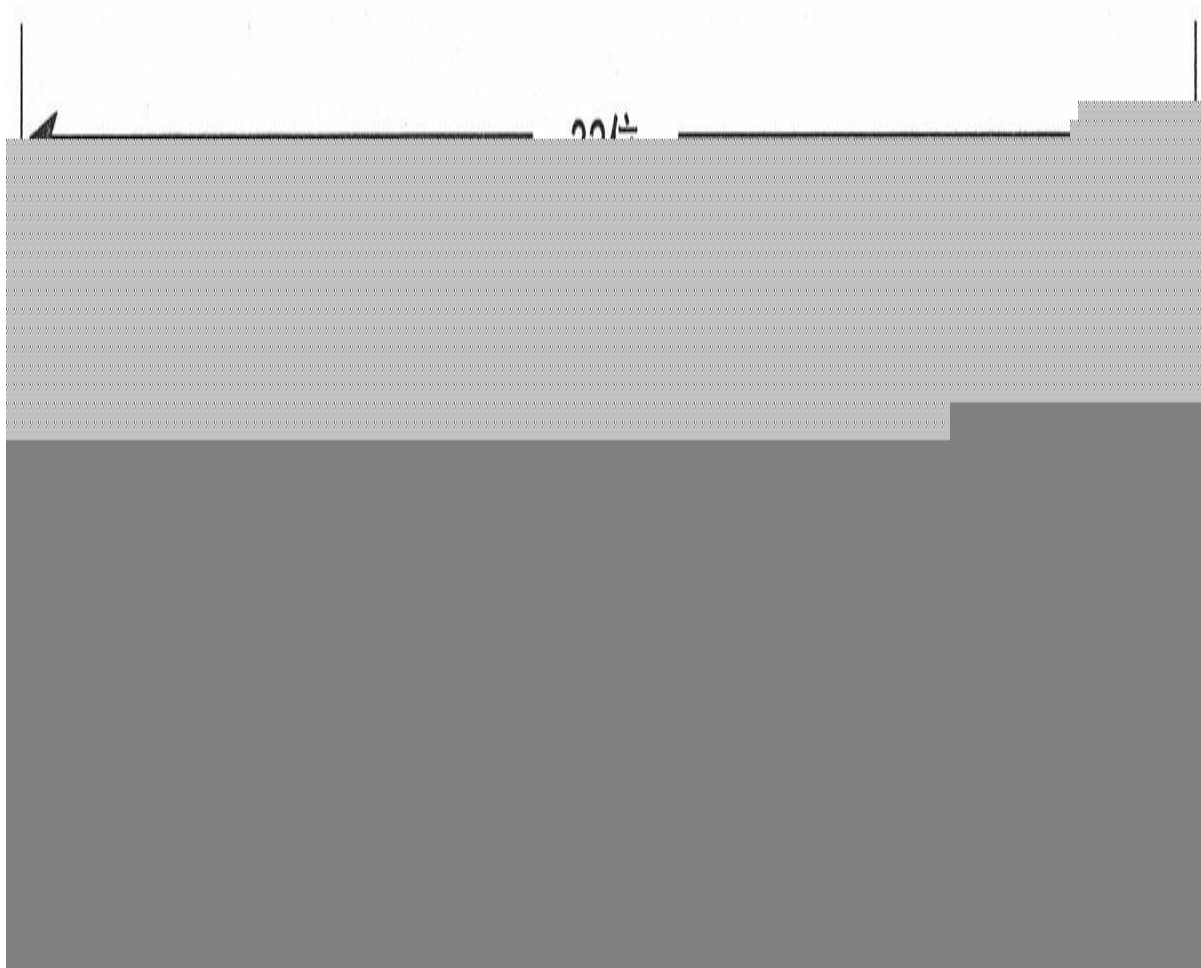
路由器Rubens的**network** 语句将匹配路由器上所有的接口。地址0.0.0.0实际上仅仅是一个占位符。在这里，反向掩码255.255.255.255才是所有操作的核心。当“可以忽略”的位占据了所有4个八位组字节时，掩码将认为是和任何地址相匹配的，并且把相应的接口指定到区域1中。这种方法提供了一种最粗略地控制接口运行OSPF的手段。

路由器Chardin是区域1和区域0之间的ABR路由器。这个事实是由在它们上面配置的**network** 语句看出的。在这里，（地址，反向掩码）对将把与主网络192.168.30.0的任何子网相连的所有接口放置到区域1中，并把与主网络192.168.20.0的任何子网相连的所有接口放到骨干区域。路由器Goya也是一台ABR路由器。在这里，（地址，反向掩码）对将只匹配两个接口上配置的具体子网。这里也要注意，骨干区域是用点分十进制来指定的。这种格式和路由器Chardin上配置的十进制格式是兼容的，它们都会使OSPF数据包格式中相应的区域字段设置为0x00000000。

路由器Matisse有一个接口192.168.10.65/26，但这个接口并不运行OSPF协议。这台路由器上的**network** 语句配置的是每个单独的接口地址，因而它的反向掩码指明所有32位都必须精确地匹配。这种方法提供了一种最精确地控制接口运行OSPF的手段。

最后，请注意虽然路由器Matisse的接口192.168.10.65/26没有运行OSPF协议，但是这个地址在数值上是该路由器上最高的IP地址。路由器Matisse的路由器ID就是192.168.10.65，参见示例8-23所示。

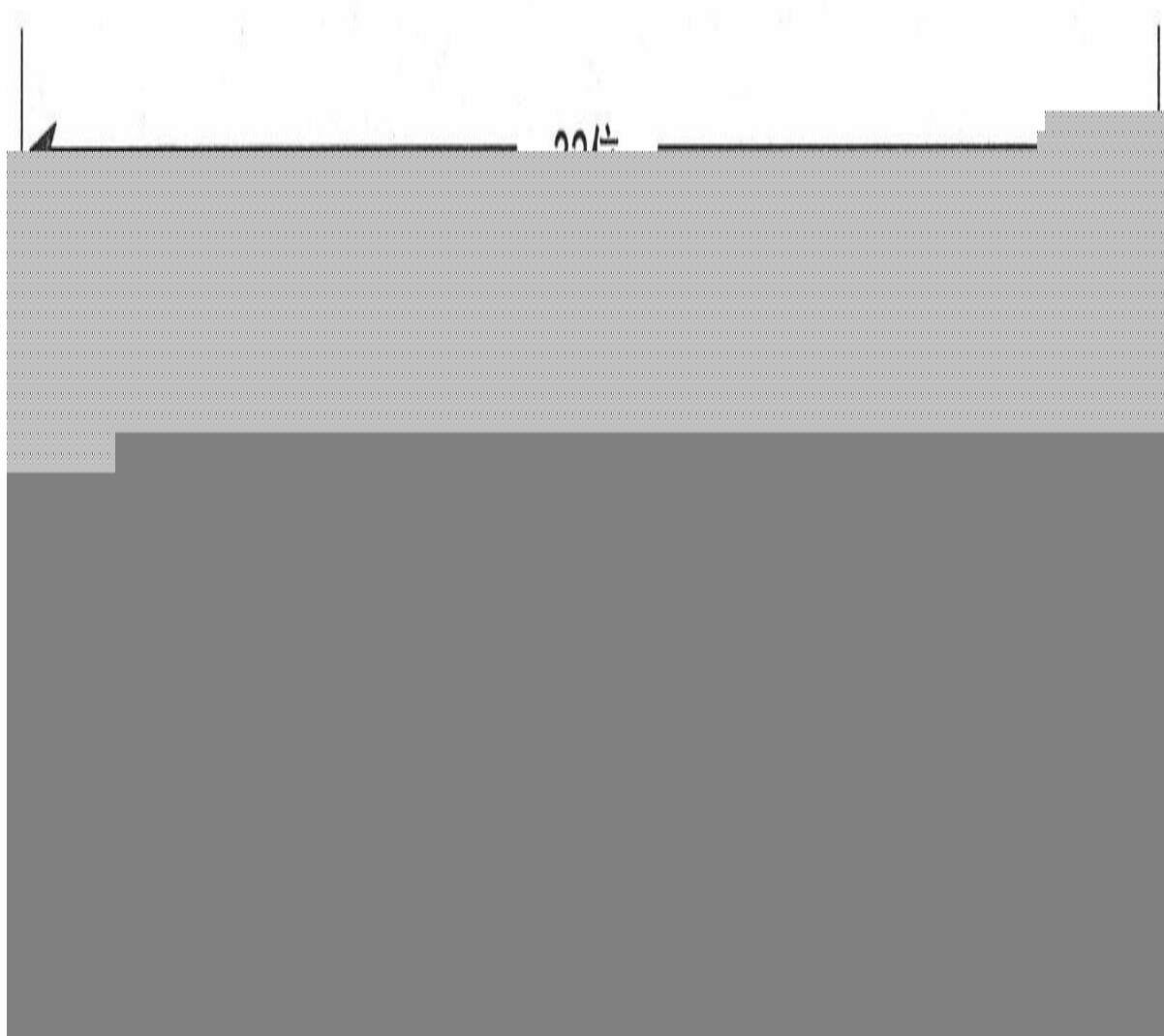
示例8-23 使用命令**show ip ospf process-id**显示了指定进程的信息



8.2.2 案例研究：使用Loopback接口设置路由器的ID

假设图8-43中的路由器Matisse已经在设备配置中心配置好了，并且被发送到了要安装的地点。在这台路由器启动的过程中，将报告它无法定位一个路由器ID，并且看上去好像在报告**network area** 命令出现配置错误，参见示例8-24。更麻烦的是，OSPF命令也不再是可运行的配置了。

示例8-24 如果找不到一个有效的**IP**地址作为它的路由器**ID**，**OSPF**将不会启动



这里出现的问题是，在路由器启动期间，其上所有的接口都是管理关闭（**administratively shutdown**）的。如果OSPF不能发现一个有效的IP地址作为它的路由器ID，那么OSPF将不会启动。再进一步，如果OSPF进程没有启动，那么随后的**network area** 命令也将是无效的。

解决这个问题的方法（在这里假定关闭所有的物理接口是有合理的原因的）是使用一个**loopback**接口（环回接口）。**loopback**接口是一个仅在软件上有意义的、虚拟的接口，并且它总是有效（**up**）的。因此，**loopback**接口的IP地址也总是有效的。

在OSPF路由器上使用**loopback**接口还有一个更普遍的原因是，这些接口允许网络管理员对路由器ID进行控制。当OSPF进程查找一个路由器ID时，OSPF将越过所有物理接口的IP地址，优先选用**loopback**接口的IP地

址，而且不管IP地址在数值上的高低顺序如何。如果路由器具有多个带IP地址的loopback接口，那么OSPF将选用在数值上最高的loopback地址。

控制路由器ID使单个OSPF路由器更加容易识别，从而使网络的管理和故障排除更加容易。路由器ID的管理通常使用以下两种方法之一：

- 单独使用合法的网络或子网地址作为路由器ID；
- 使用一段“伪造”的IP地址段。

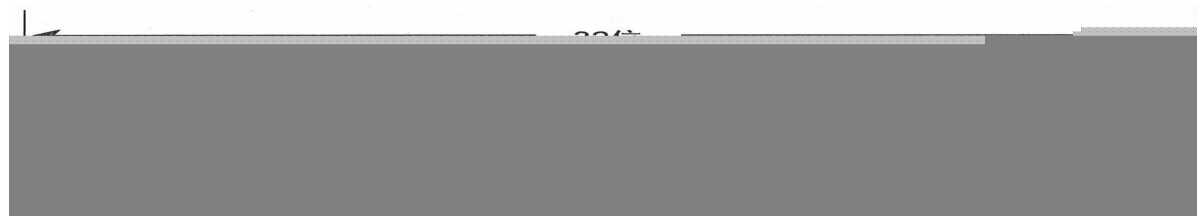
第一种方法的缺点是需要使用合法的网络地址空间。第二种方法可以节省合法的地址，但是有一点要记住，在一个网络里伪造的地址在另一个网络里可能是合法的地址。只要你记得这些地址不是合法的地址，那么使用一些简单、易于识别的地址，例如1.1.1.1、2.2.1.1等将是比较好的做法。必须要小心的是，伪造的地址千万不能泄漏到公共的Internet网络上去。

在前面一节的配置中使用loopback地址进行修改参见示例8-25～示例8-28。

示例8-25 路由器**Rubens**的配置中增加了一个**loopback**接口



示例8-26 路由器**Chardin**的配置中增加了一个**loopback**接口



示例8-27 路由器**Goya**的配置中增加了一个**loopback**接口

示例8-28 路由器Matisse的配置中增加了一个loopback接口

对于这个例子，网络地址192.168.50.0独自使用来作为路由器ID。因此，在这个网络中，路由器ID可以容易地和其他IP地址区分开来。

这里要注意的第一件事情是，在这个配置中loopback地址所使用的地址掩码：每一个掩码都配置成一个主机地址。这一步其实不是必要的，因为OSPF会把一个loopback接口作为一个末梢网络来看待。无论（地址，掩码）对配置成什么，loopback接口的地址都将被作为一条主机路由来通告。主机掩码仅仅用来保持一种整齐的格式，并且用来反映所通告的地址的一种方式。

然而，第二个需要引起注意的地方和第一个有点不相关。请记住，OSPF协议虽然使用一个接口的IP地址作为它的路由器ID，但是并不一定需要在这个接口上运行OSPF。事实上，OSPF所通告的loopback地址不过是创建了一个不必要的LSA而已。在上面一个例子的显示中要注意，那里的**network area** 语句并没有涉及到loopback地址。事实上，路由器Rubens上的配置不得不更改。路由器Rubens在前面例子中的命令 **network 0.0.0.0 255.255.255.255 area 1** 应该已含有loopback地址。

另外，为了对网络的管理和故障排除有所帮助，使用loopback接口也可以使一个OSPF网络更加稳定。在一些早期的IOS软件版本中，如果一个作为路由器ID的物理接口出现了硬件故障，[\[27\]](#)或者这个接口被管理关闭（administratively shutdown），或者这个IP地址被无意删除，那么OSPF进程将必须获取一个新的路由器ID。因此，路由器必须过早地老化和泛洪扩散它原来的LSA，并且要泛洪扩散包含新的路由器ID的

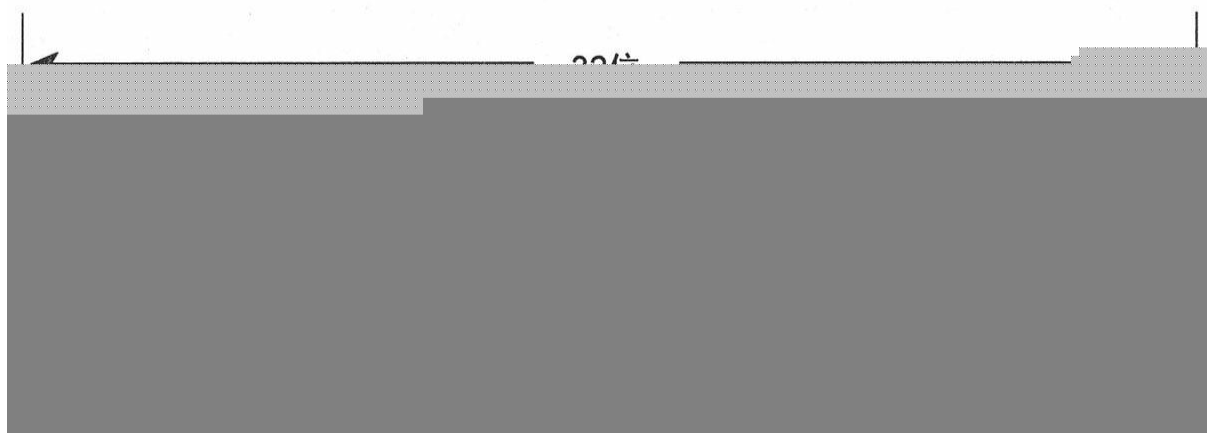
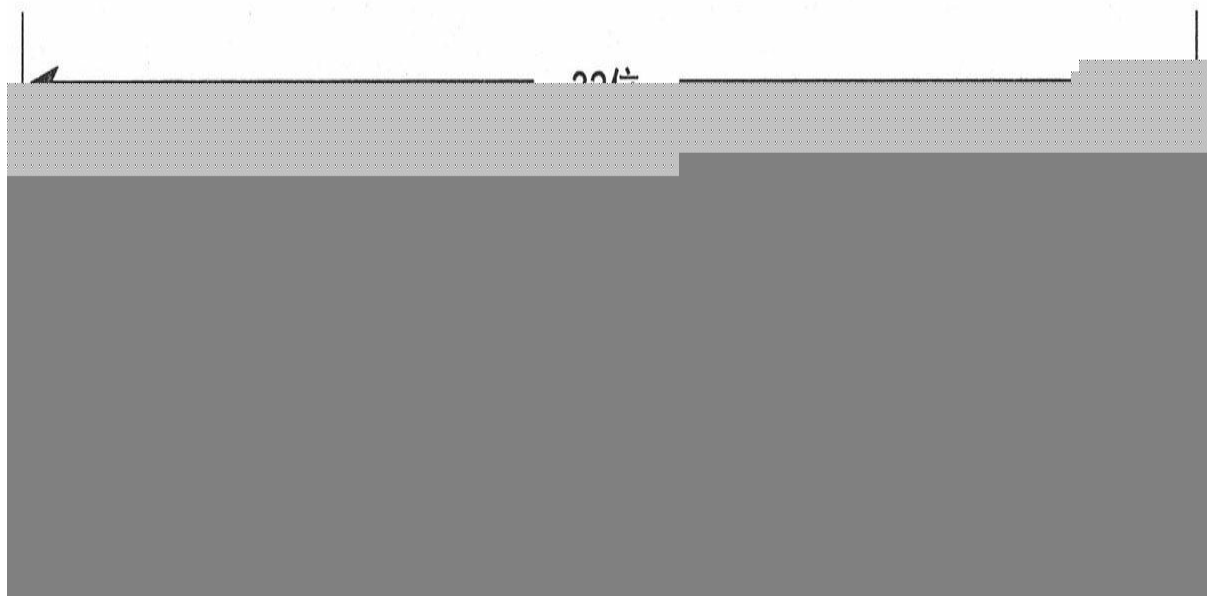
LSA。loopback接口却没有硬件上的故障问题。目前的IOS软件版本是获取路由器ID，如果路由器ID相关联的接口失效或它的IP地址被删除，那么这台路由器只有在它被重新启动或重新运行OSPF进程后才能获取路由器ID。

另一种可行的做法是，使用OSPF命令**router-id** 手工为该路由器指定一个路由器ID。当使用loopback接口作为路由器ID时，你可以任意选择一个IP地址作为**router-id** 命令的值。如果使用该命令配置一个路由器ID，这个命令的值将在重启OSPF进程或重启该路由器时成为路由器ID。然而，在路由器被重新启动的时候，OSPF进程还没运行之前，必须要有一个在线的带有IP地址的接口。在OSPF进程运行后，使用**router-id** 命令指定的地址将成为路由器ID。

8.2.3 案例研究：域名服务查询

loopback地址由于可以提供预先设计好的路由器ID，从而使一个OSPF网络的管理和故障排除变得很简单。为了进一步得到简化，可以把路由器ID记录到一个域名服务（DNS）数据库中。在路由器上可以配置域名服务，使路由器向一台域名服务器请求相关“地址到名称”的映射，或称为反向DNS查找，从而可以通过显示路由器的名称来代替路由器的ID，参见示例8-29所示。

示例8-29 OSPF可以配置成利用DNS来实现某些show命令中路由器ID到名称的映射



路由器Goya可以使用示例8-30中的配置来执行DNS的查找。

示例8-30 配置路由器Goya执行DNS查找



第一个命令用来指定一个DNS服务器的地址，第二个命令用来启动OSPF进程执行DNS查找。在一些实际案例中，一台路由器是通过一个接口地址来识别的，而不是路由器ID。这种情况下，可以为路由器的这个接口增加一个条目到DNS数据库中，例如rubens-e0。这样做可以使路由器通过这个接口的名字来识别，这与路由器ID不同。

在这个例子中使用的域名服务器的地址是不属于图8-43中显示的某一个子网的。到达这个网络的方法将是下一个案例研究的主题。

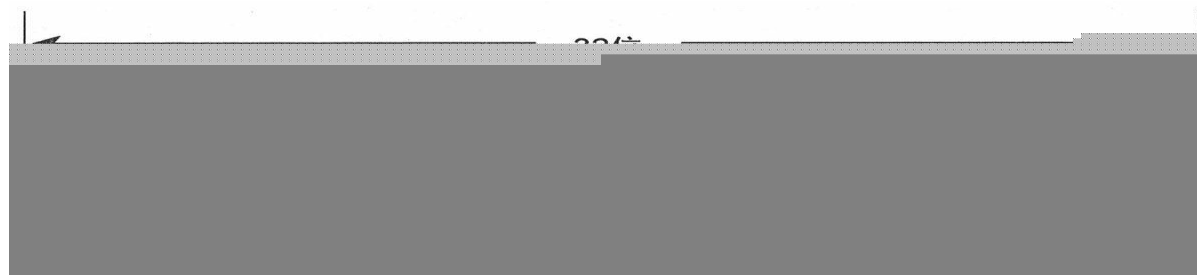
8.2.4 案例研究：OSPF和辅助地址

在一个OSPF的环境中，辅助地址的用法有以下两个相关的规则：

- 只有在主网络或子网也运行OSPF协议的时候，OSPF才会通告一个辅助的网络或子网。
- OSPF将把辅助地址看作是末梢网络（这些网络上没有OSPF邻居），从而不会在这些网络上发送Hello数据包。因此，在辅助网络上也就无法建立邻接关系。

图8-44中显示了一台DNS服务器，并另外增添了一台路由器连接到路由器Matisse的FA0/0接口上。这台DNS服务器和新加的路由器都放在子网172.19.35.0/25中，并给路由器Matisse的FA0/0接口分配一个辅助地址172.19.35.15/25（参见示例8-31）。

示例8-31 路由器Matisse配置了一个辅助地址



根据这个配置，路由器Matisse将向它的邻居路由器通告子网172.19.35.0/25。但是，如果关于主地址192.168.10.33的**network area** 语句被删除了的话，那么子网172.19.35.0/25也将不再被通告。

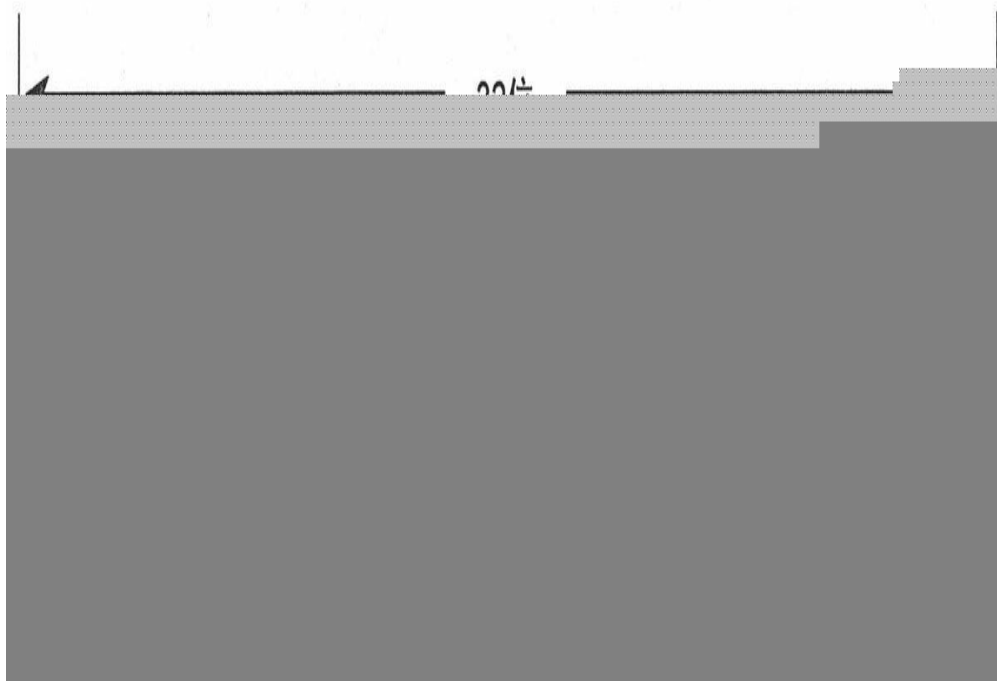


图8-44 路由器**Dali**和**DNS**服务器都不是**OSPF**域的一部分，并且它们是通过一个辅助地址连接到路由器**Matisse**上的

由于路由器**Matisse**是通过一个辅助地址和子网172.19.35.0/25相连的，因此它不能和这个子网上的任何路由器建立邻接关系，如图8-45所示。但是，**DNS**服务器使用了路由器**Dali**作为它的缺省网关。因此，路由器**Matisse**和路由器**Dali**之间必须能够互相转发数据包。

到目前为止，对于上述的网络可以得出以下的结论：

- 子网172.19.35.0/25正在被通告到**OSPF**域中；一个目的地址是172.19.35.2的数据包将被转发到路由器**Matisse**的FA0/0接口，并且从那里直接转发到**DNS**服务器，参见示例8-32所示。
- 由于**DNS**服务器必须发一些回复数据包到与它不在同一网段的网络，因而它会根据路由选择发送这些回复到路由器**Dali**。
- 路由器**Dali**和路由器**Matisse**之间并不交换路由选择信息，因此它将不知道怎样到达**OSPF**域内的网络。

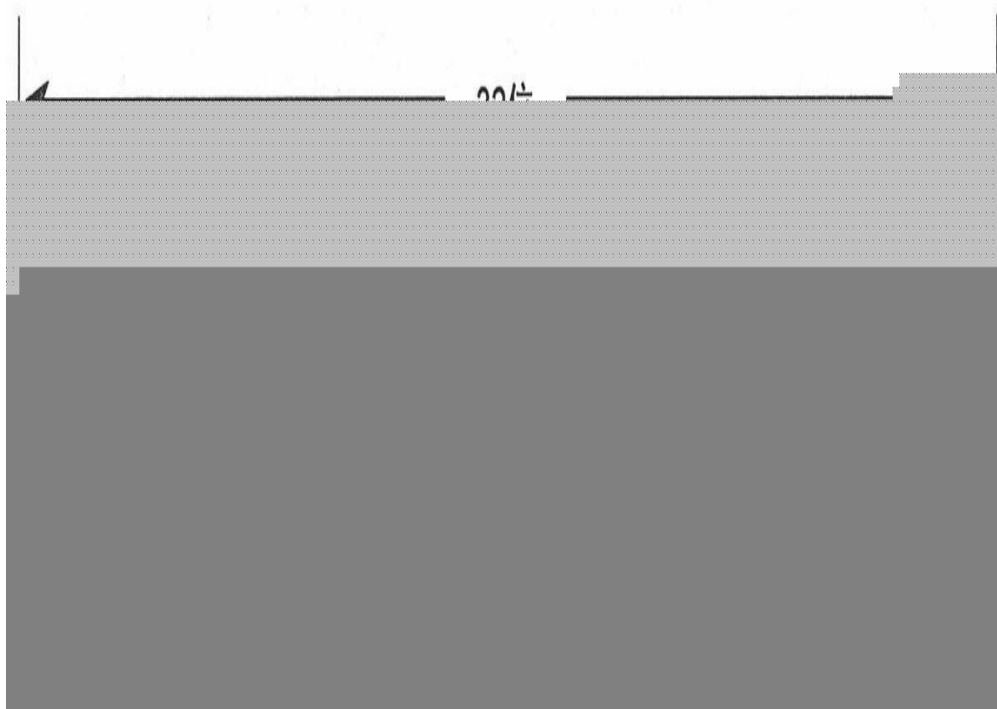
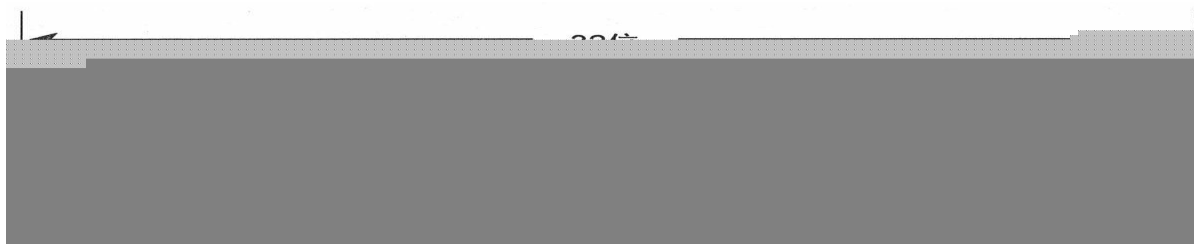


图8-45 这台协议分析仪显示的信息是从与路由器**Matisse**、**Dali**和**DNS**服务器相连的网络上捕获得。小一点的窗口中显示出**Hello**数据包只能由路由器**Matisse**的主地址**192.168.10.33**发送。大一点的窗口显示了一个**Hello**数据包的解码信息

示例8-32 在路由器**Matisse**的**ARP**缓存中记录了**DNS**服务器的**MAC**地址标识，这表明该**DNS**服务器是可以直接到达的。但是，如果到达**DNS**服务器的数据包必须通过路由器**Dali**路由转发才能到达的话，那么在这个缓存中，这台**DNS**服务器和路由器**Dali**的**MAC**地址将应该都是**0000.0c0a.2aa9**



因此，这里需要一步去“连通一条电路”，来告诉路由器**Dali**怎样才能到达**OSPF**域内的网络。这一步可以通过配置一条静态路由很容易地完成：

Dali (config) #ip route 192.168.0.0 255.255.0.0 172.19.35.15

这里需要注意的是，静态路由是无类别的路由，因而它可以使用超网的条目来匹配OSPF自主系统内的所有地址。

在这个例子当中，路由器Matisse并不是一台ASBR路由器。这是因为，虽然它将数据包发送到OSPF自主系统外部的目的地，但是它并没有接受外部目的地的任何信息，因此，也没有始发任何类型5的LSA通告。

图8-46中显示了通过路由器Dali到达的一组新的目的地址。路由器Matisse现在就必须变成一台ASBR路由器，并且要将这些路由通告到OSPF域内。但是，它首先必须学习到这些路由。这个工作可以通过配置一系列静态路由或者运行一个在辅助网络上通信的路由选择协议来完成。无论在上述哪种情况下，路由器都必须重新分配这些路由到OSPF域中。

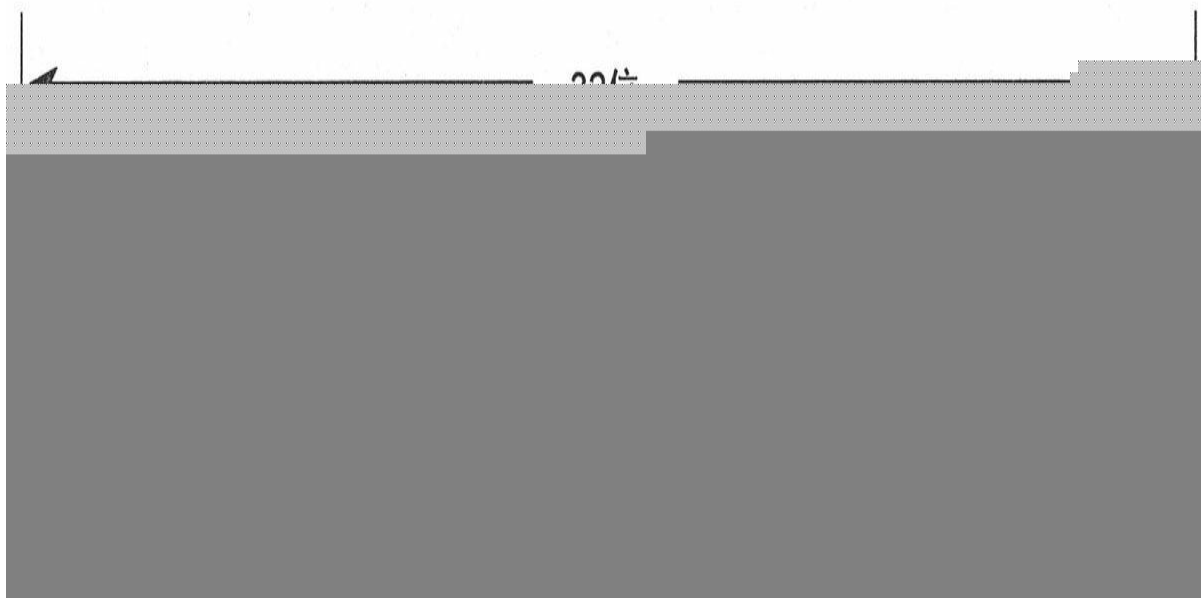
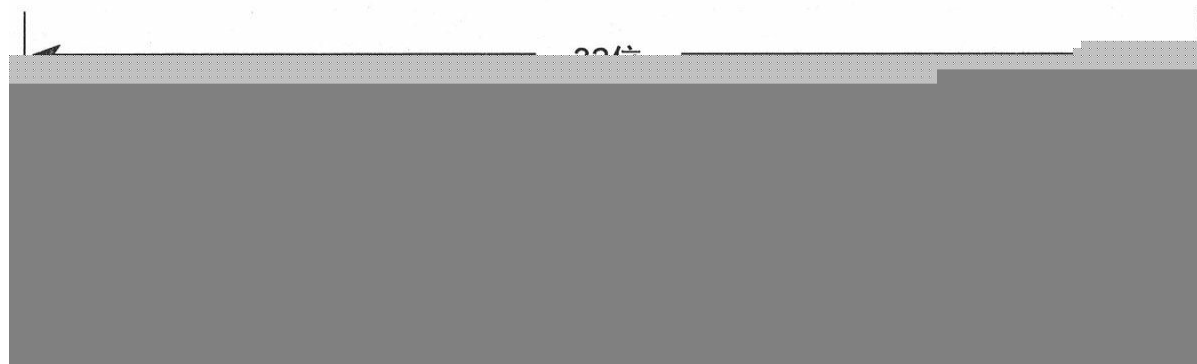


图8-46 OSPF自主系统必须学习到这些通过路由器Dali可达的目的地址，但是路由器Matisse到达路由器Dali的辅助地址妨碍了这两台路由器共享OSPF信息

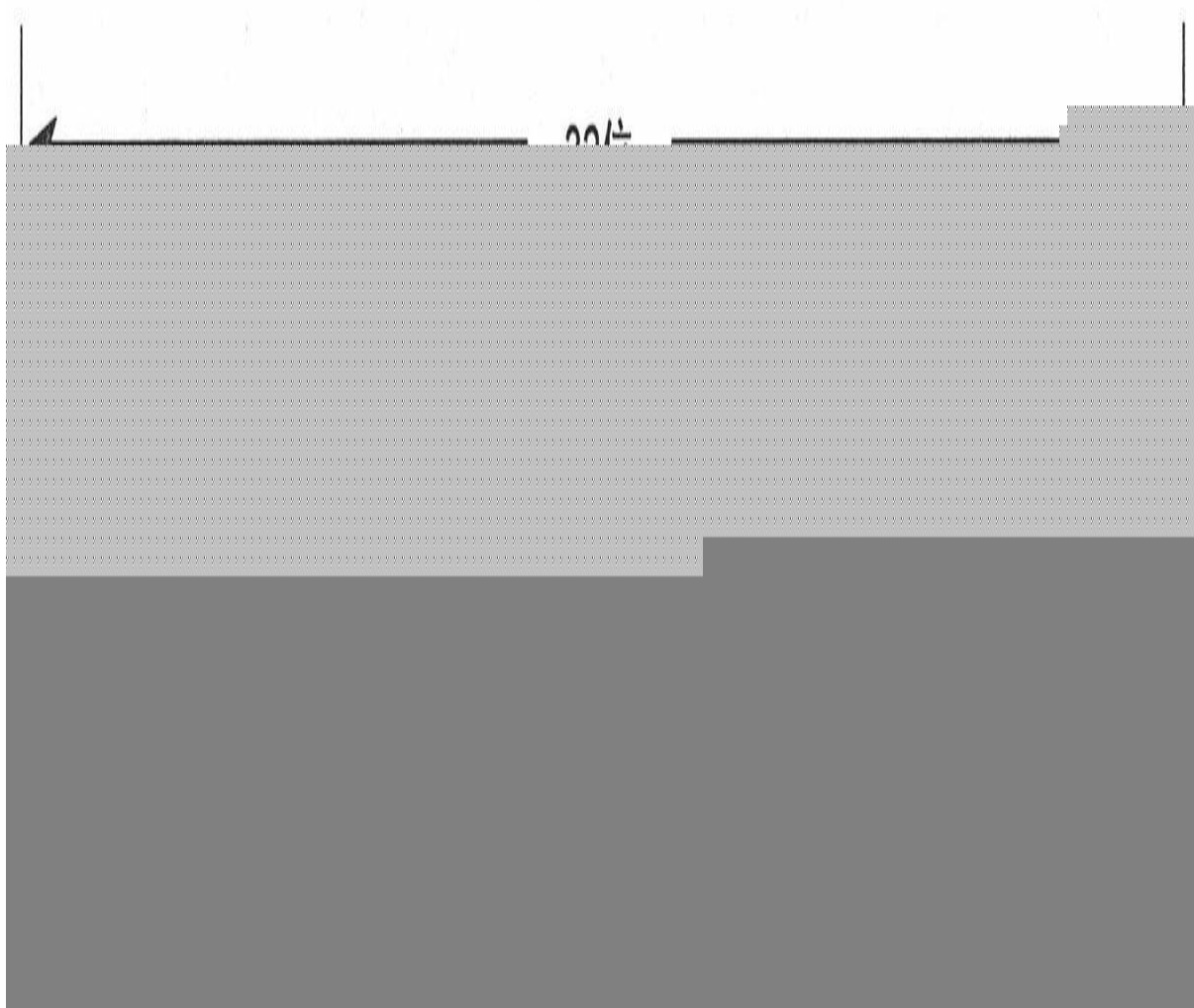
RIP协议在辅助地址上使用没有什么困难。因此，在路由器Matisse上可以选用RIP协议作为与路由器Dali的通信。路由器Matisse的配置参见示例8-33。

示例8-33 在路由器Matisse的配置中增加了RIP

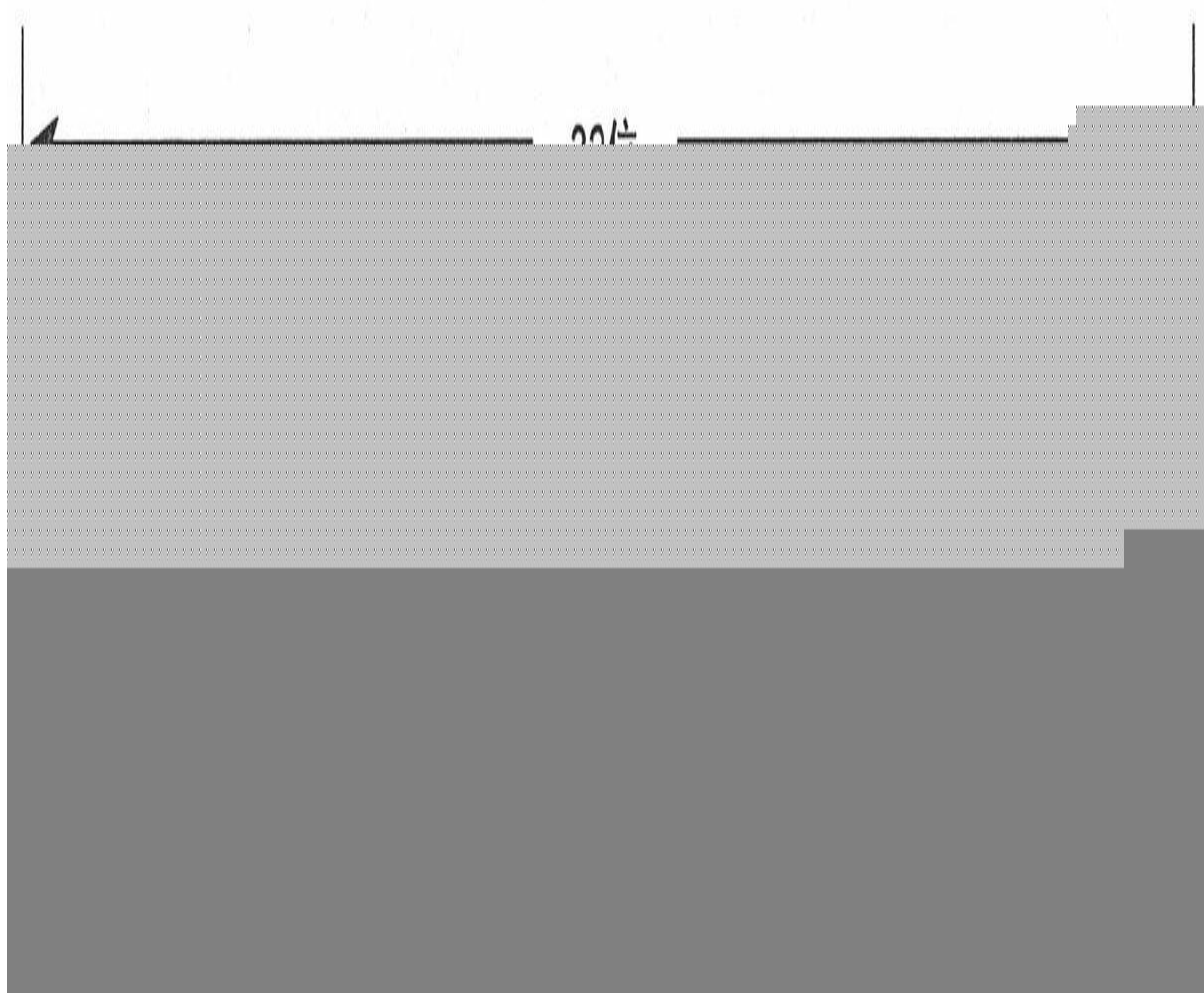


上述配置可以在FA0/0接口的辅助网络上启用RIP协议，它可以让路由器Matisse从路由器Dali那里学习到路由，参见示例8-34所示。这些路由重新分配到OSPF域中（此时这些路由已不再运行在辅助地址上了），并且给它们指定一个OSPF的度量代价为10，这是通过命令**redistribute rip metric 10 subnets**来实现的。关于路由重新分配的更加详细的介绍请参考第11章。在示例8-35中，通告到OSPF域内的路由是使用缺省的外部类型度量的，即外部类型2（E2）。注意观察在路由器Rubens上，这些路由的代价仍然是10。路由器Matisse将使用类型5的LSA通告这些外部的目的地址，并使自己成为一台ASBR路由器（参见示例8-36所示）。

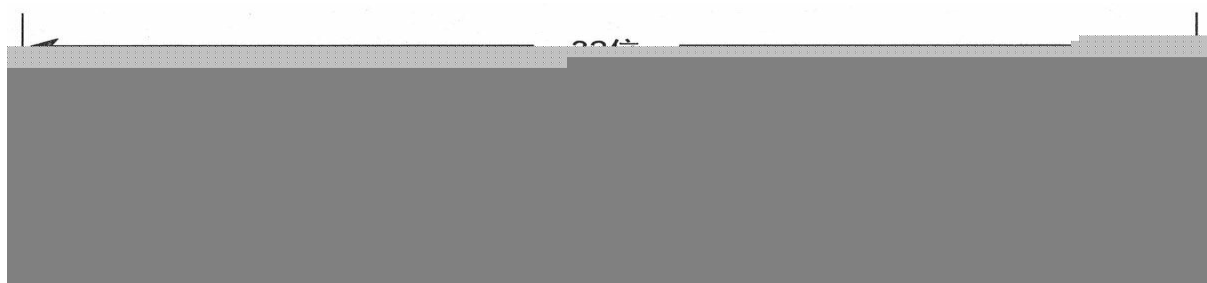
示例8-34 路由器Dali通过RIP协议将它的路由选择信息传递给路由器Matisse



示例**8-35** 通过**RIP**协议学习到的路由将作为路径类型**E2**的路由重新分配到**OSPF**自主系统当中



示例8-36 路由器**Matisse**（RID=192.168.50.4）现在变成了一台**ASBR**路由器，这是因为它正在始发自主系统外部**LSA**来通告外部路由



8.2.5 案例研究：末梢区域

由于在图8-46中的区域1里面没有始发类型5的LSA，因而它可以配置成一个末梢区域。注意，当一个相连的区域被配置成一个末梢区域时，路

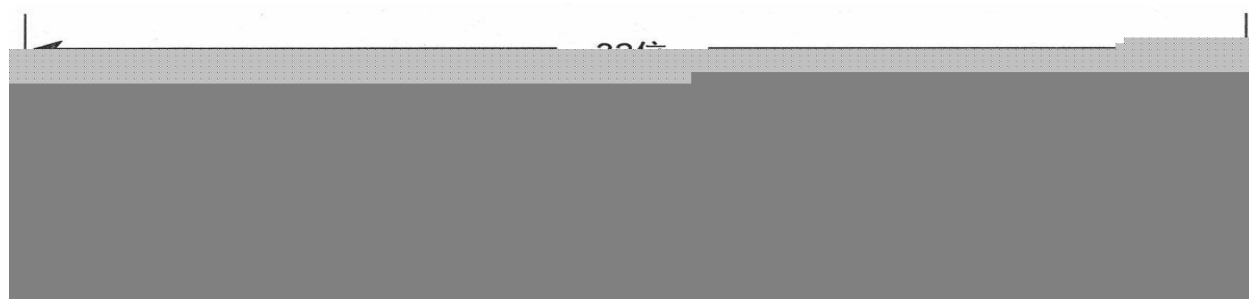
由器始发的Hello数据包进入那个区域后，它的可选字段中的E位将会设置为0，即E=0。其他没有同样配置的所有路由器收到这些Hello数据包后将丢弃这些数据包，并且不能在这些路由器之间建立邻接关系。即使已经存在一个邻接关系，它也会被阻断。因此，如果需要把一个正在运行的区域重新配置成一个末梢区域的话，应该计划好路由阻断的时间；因此在这期间路由将被阻断，一直到所有的路由器都重新配置后才能恢复。

一个末梢区域的配置可以通过在OSPF进程中增加命令area stub来完成，具体配置参见示例8-37和示例8-38。

示例8-37 在路由器**Rubens**上把区域**1**配置为末梢区域

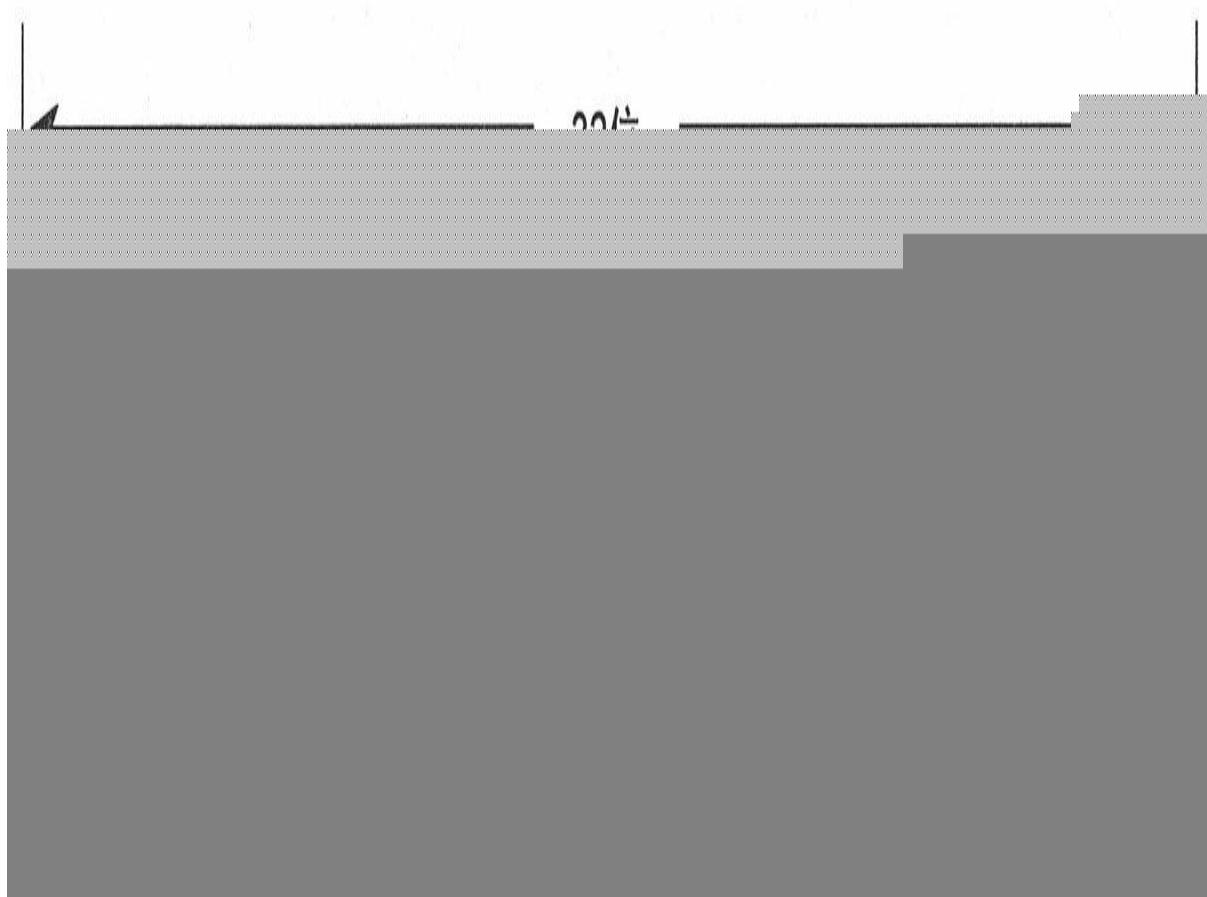


示例8-38 在路由器**Chardin**上把区域**1**配置为末梢区域

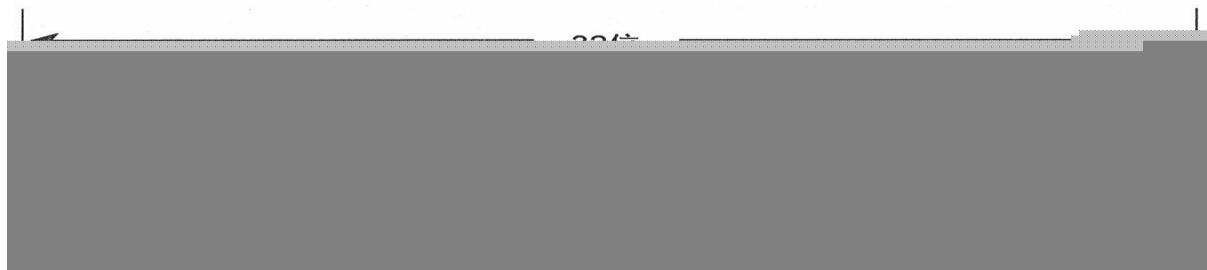


比较路由器Rubens在配置成末梢区域前（参见示例8-39）、后（参见示例8-40）的链路状态数据库，可以显示出所有的自主系统外部LSA和ASBR汇总LSA都已经从这个数据库中清除。在这种情况下，数据库的大小也缩减了50%。

示例8-39 在区域**1**配置成末梢区域前，路由器**Rubens**的数据库总共包括**14条LSA**

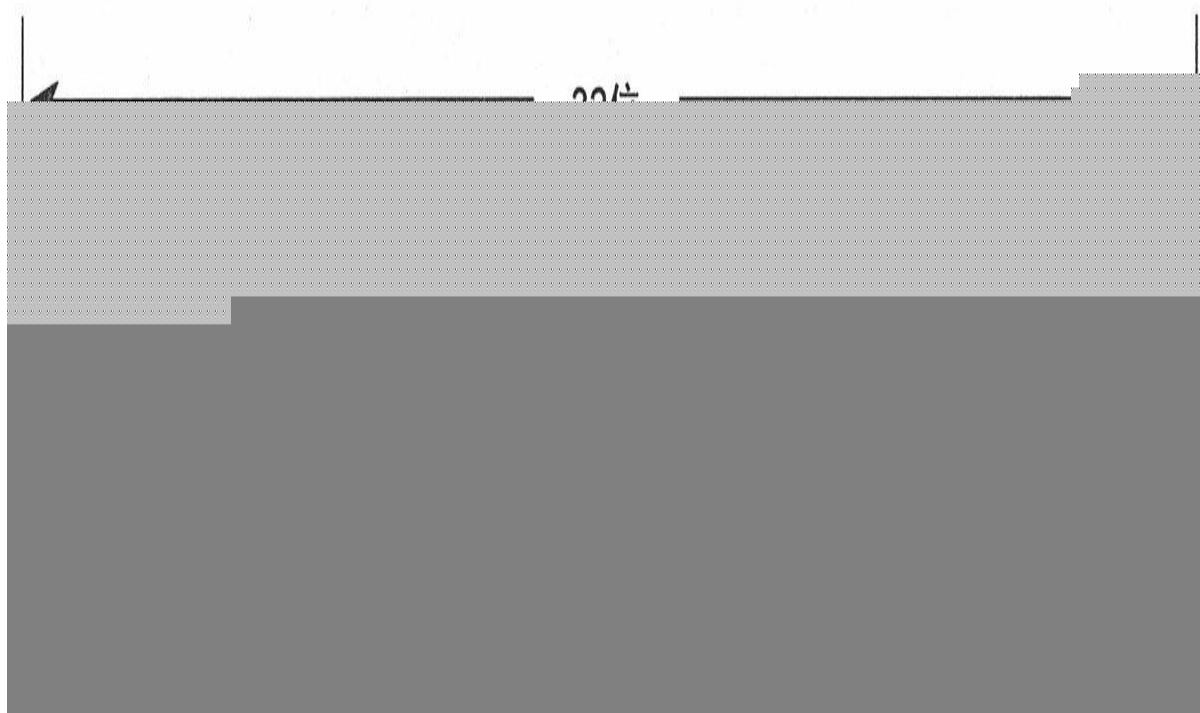


示例840 在配置成末梢区域后，使路由器**Rubens**从它的数据库中清除了**7**条类型**5**的**LSA**通告和**1**条类型**4**的**LSA**，而只增加了一条通告缺省路由的类型**3**的**LSA**



当一个末梢区域和一台ABR路由器相连时，路由器将会通过一条网络汇总LSA自动地通告一个缺省路由（目的地址是0.0.0.0）到这个区域。在示例8-40中所示的数据库汇总信息表明了这个额外的类型3的LSA。路由器Rubens的路由表（示例8-41）中的最后一个条目显示了这条缺省路由是由路由器Chardin通告的。

示例8-41 路由器**Rubens**的路由表显示，所有的外部路由都被清除（请将这里所显示的和示例8-35显示的相比较），但增加了一条缺省路由



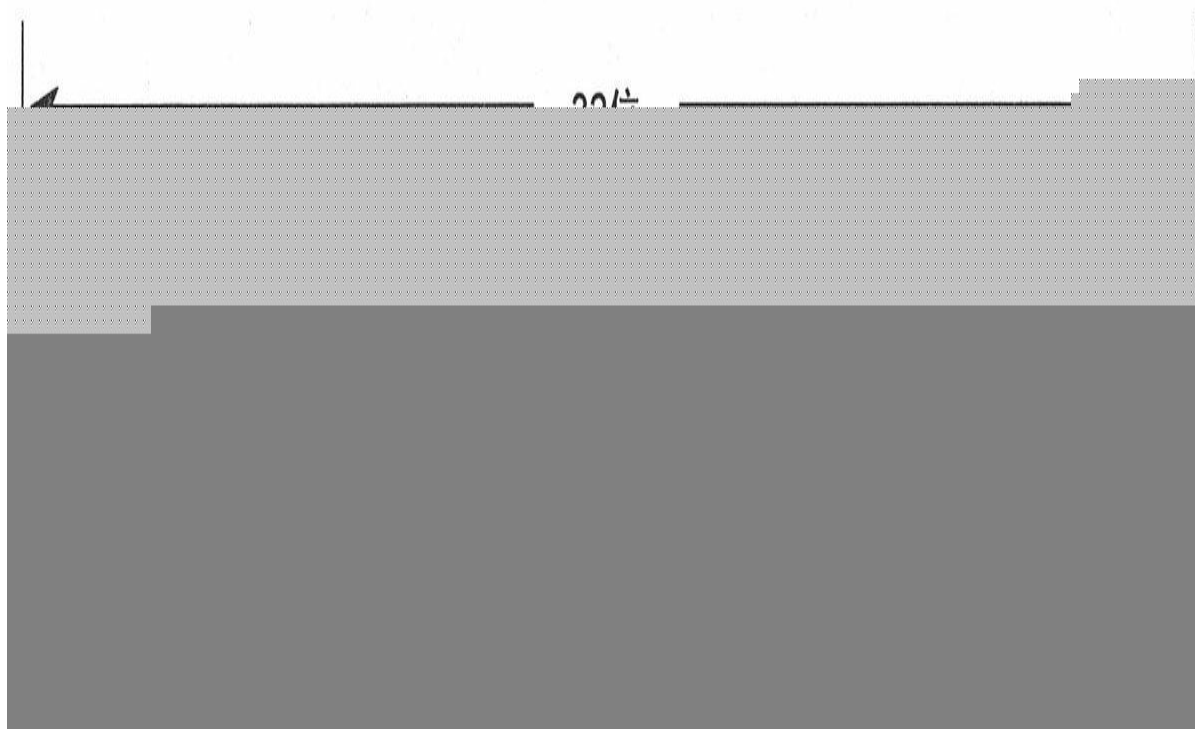
ABR路由器将通告代价为1的缺省路由。在路由器Rubens和路由器Chardin之间的串行链路的代价是64，在示例8-41中显示了路由器Rubens到达缺省路由的总代价将是 $64+1=65$ 。这里的缺省代价可以通过命令 **area default-cost** 来改变。例如，路由器Chardin可以配置一条代价为20的缺省路由，配置参见示例8-42所示。

示例8-42 路由器Chardin的配置设置了缺省路由的代价



结果，从示例8-43中可以看出，这条缺省路由的代价增大了—— $64+20=84$ 。在这里改变缺省路由的代价没有什么实际意义，但是在多于一台ABR路由器的末梢区域里将可能会比较有用。通常情况下，每一台内部路由器都只会选择代价最低的缺省路由。通过操纵和控制所通告的代价，网络管理员可以促使所有的内部路由器都使用同一台ABR路由器。关于第二台ABR路由器，可以通告比较高的代价，而使它仅仅在第一台ABR路由器失效时才被使用。

示例8-43 路由器**Rubens**的路由表反映了更改缺省路由的代价后的结果



8.2.6 案例研究：完全末梢区域

完全末梢区域的配置可以通过在命令**area stub** 的末端增加关键字**no-summary** 来实现。这一步的配置操作只有在ABR路由器上才是必需的，在内部路由器上使用标准的末梢区域配置就可以了。把图8-46的区域1配置成一个完全末梢区域，路由器Chardin上的配置参见示例8-44。

示例8-44 在路由器**Chardin**上把区域1配置成一个完全末梢区域



在示例8-45中，可以看出路由器Rubens的数据库中LSA的数量已经减少到3条了。在示例8-46中显示了它的路由表。

示例8-45 改变区域1成为一个完全末梢区域，消除了类型3的LSA之

外的所有**LSA**（缺省路由）

示例8-46 在完全末梢区域中的路由表将只包含区域内的路由和缺省路由

8.2.7 案例研究：NSSA区域

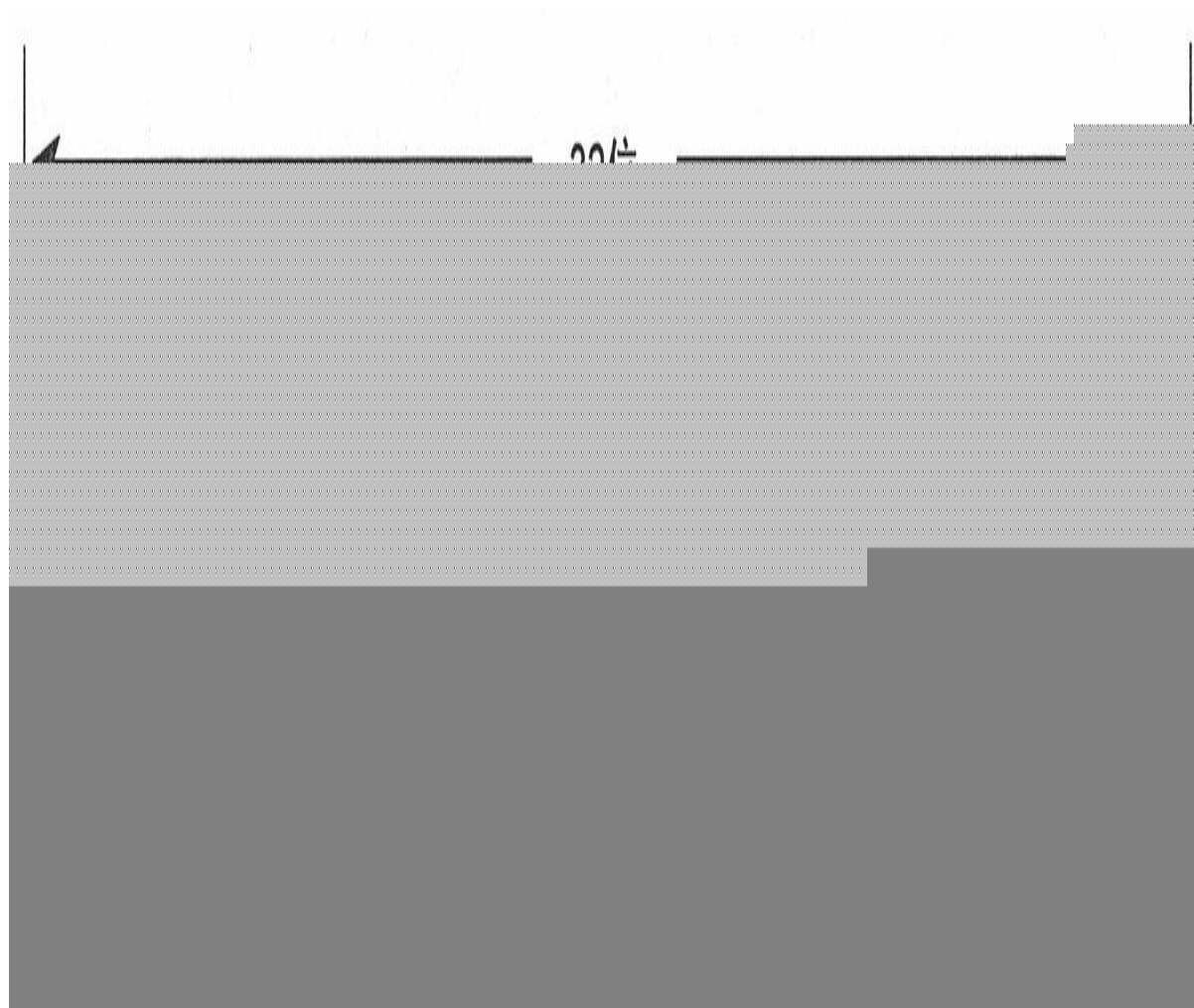
在前面的8.2.4小节中，讨论到路由器Matisse接受了通过RIP从路由器Dali学习到的路由，并把这些路由重新分配到OSPF域中（请参见图8-46）。这一步操作使路由器Matisse成为一台ASBR路由器，更进一步来说，也使区域192.168.10.0无法满足成为一个末梢区域或完全末梢区域的条件。然而，这里并不需要AS外部LSA从骨干区域通告到这个区域。因此，可以把区域192.168.10.0配置成一个NSSA，在路由器Matisse上的配置见示例8-47。

示例8-47 路由器Matisse把区域192.168.10.0配置成一个NSSA

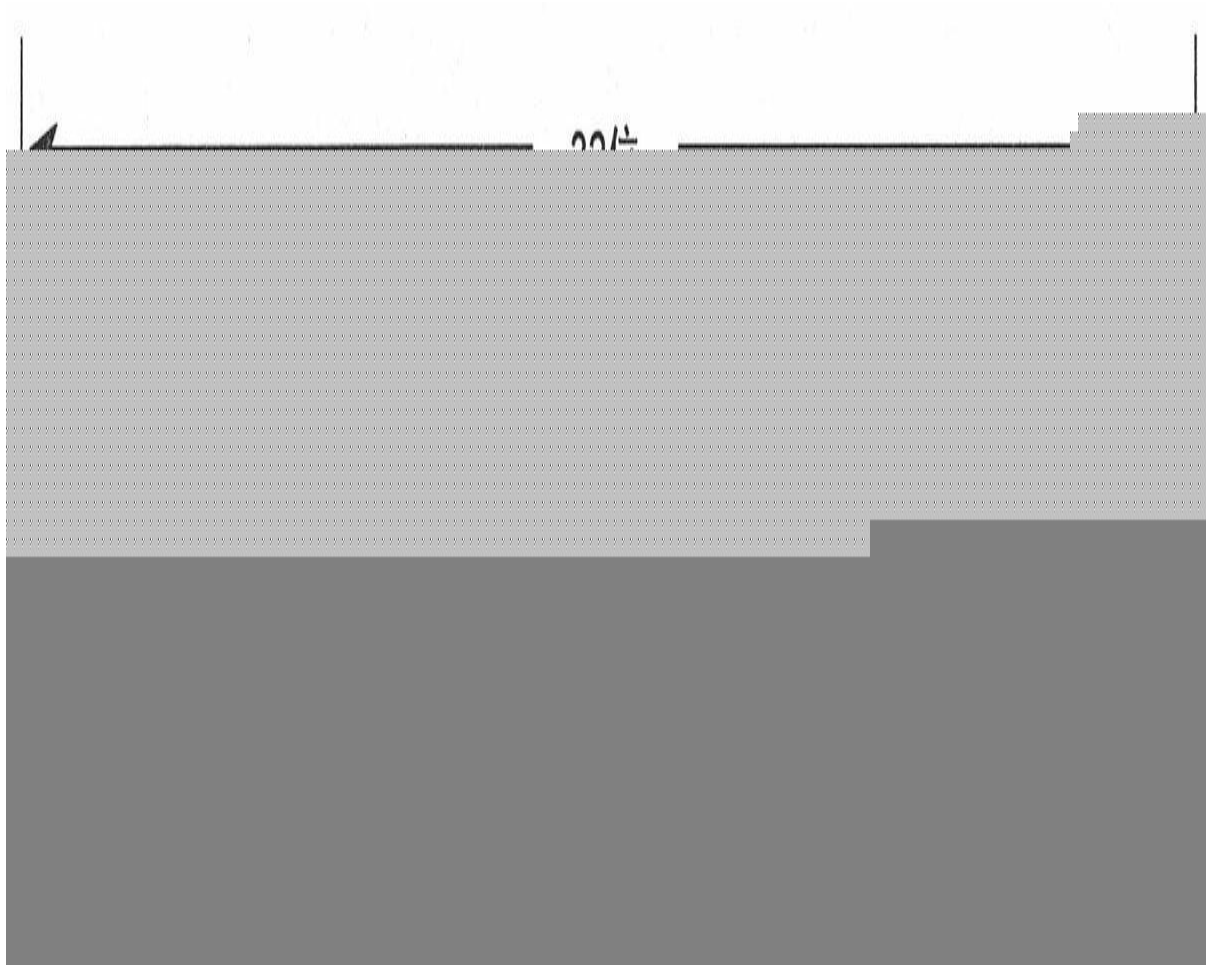
这里显示的**area nssa** 语句可以同样地配置在路由器Goya上。由于路由器Goya是一台ABR路由器，因此它将把与NSSA区域相连的接口收到的类型7的LSA转换成类型5的LSA。这些转换过的LSA将泛洪扩散到整个骨干区域中去，从而也泛洪扩散到其他的区域中去。比较路由器Goya和路由器Chardin的路由表，可以看出路由器Goya把外部路由标记成NSSA

[28]（如示例8-48所示）。而路由器Chardin把外部路由标记成E2（如示例8-49所示），这表明它们是从类型5的LSA学到的。

示例8-48 从路由器Matisse学到的外部路由在路由器Goya上被标记成NSSA路由

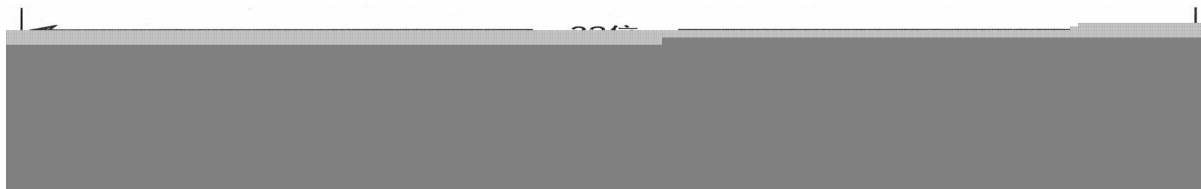
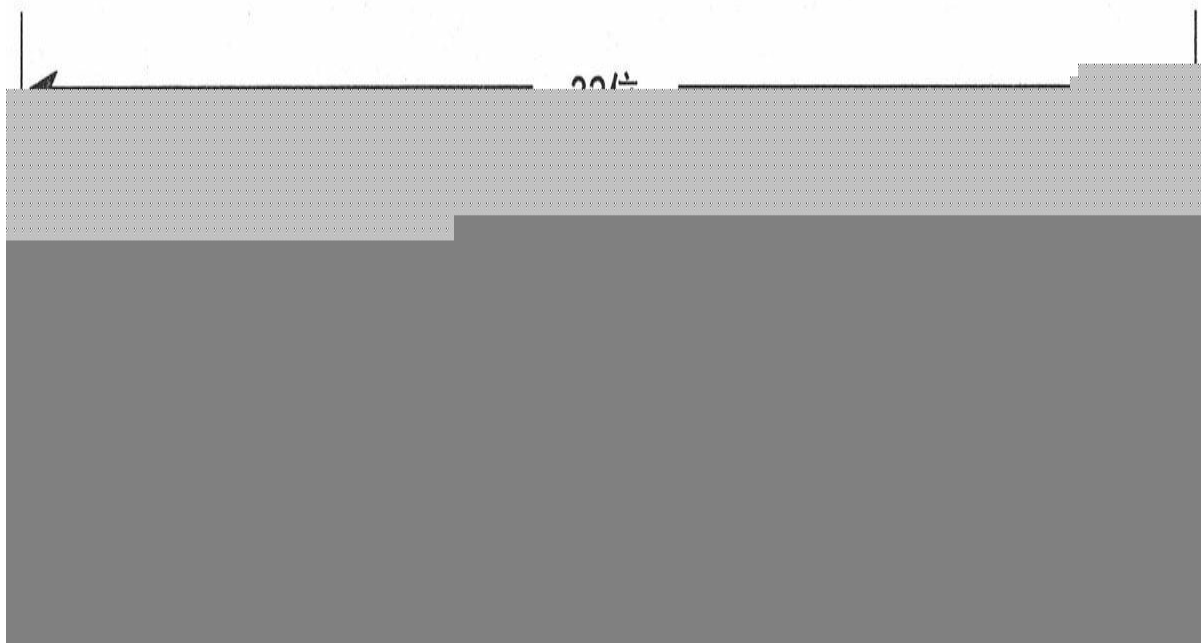


示例8-49 路由器Chardin把同样的路由标记成E2类型，表明它们是从自主系统外部LSA学习到的



这个转换也可以通过检查路由器Goya的链路状态数据库来观察。在示例8-50中，数据库中同时包含了相同外部路由的类型7和类型5的LSA。只不过类型7的LSA是始发于路由器Matisse的，而类型5的LSA是始发于路由器Goya的。

示例8-50 路由器Goya的链路状态数据库表明了来自路由器Matisse（192.168.50.4）的类型7 LSA已经被Goya（192.168.50.3）转换成类型5的LSA了



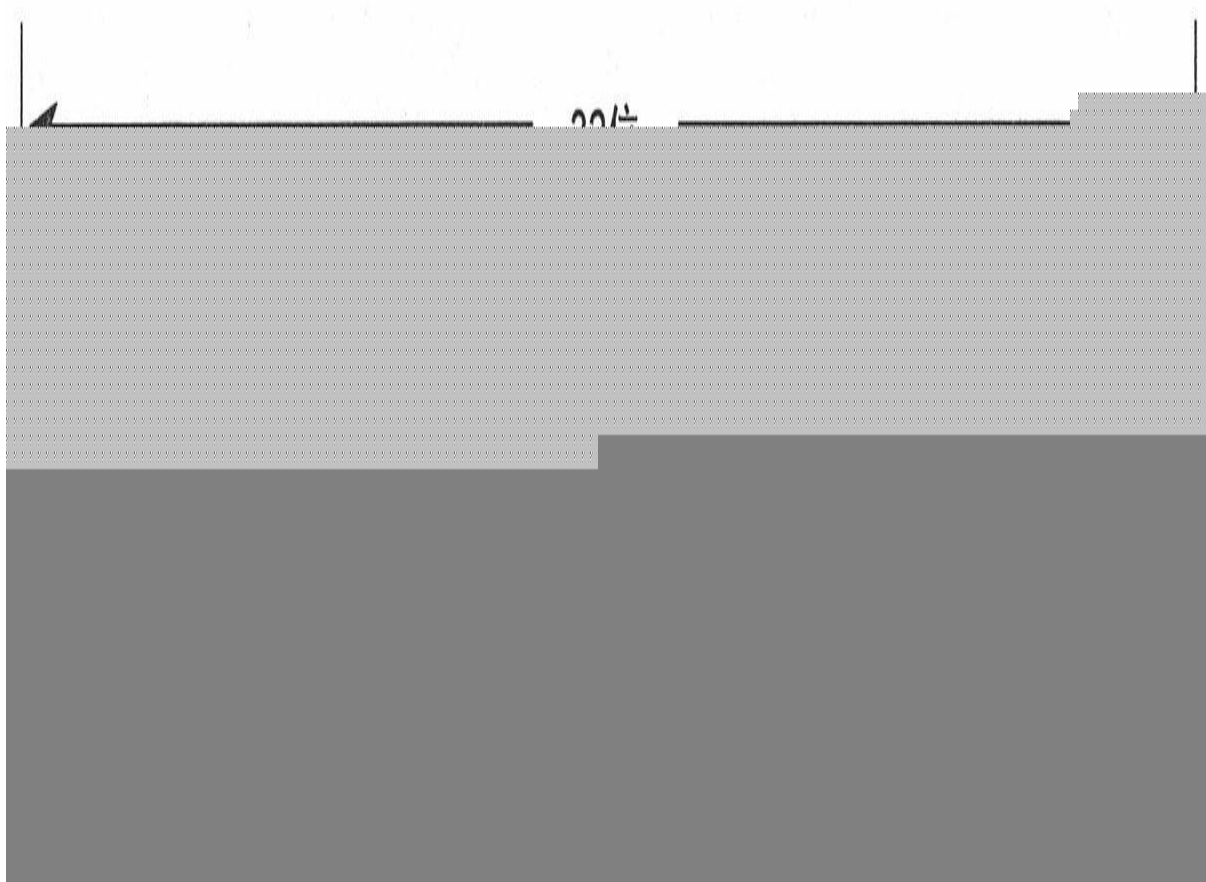
对于ABR路由器来说还有几个可用的配置选项。第一个是**no-summary**选项，可以和命令**area nssa**一起用来阻塞类型3和类型4的LSA泛洪扩散到NSSA中。这个配置可以使区域192.168.10.0变成一个名字有点怪异的区域——“完全非纯末梢区域”（totally stubby not-so-stubby area）。这时，路由器Goya上的配置应该如示例8-51所显示的。

示例8-51 路由器**Goya**把区域**192.168.10.0**配置成一个完全非纯末梢区域



路由器Matisse的路由表参见示例8-52所示，可以看出所有的区域间路由都被消除了，而另外增加了一条由路由器Goya通告的缺省路由。

示例8-52 所有的区域间路由都被一条到达**ABR**的缺省路由替代



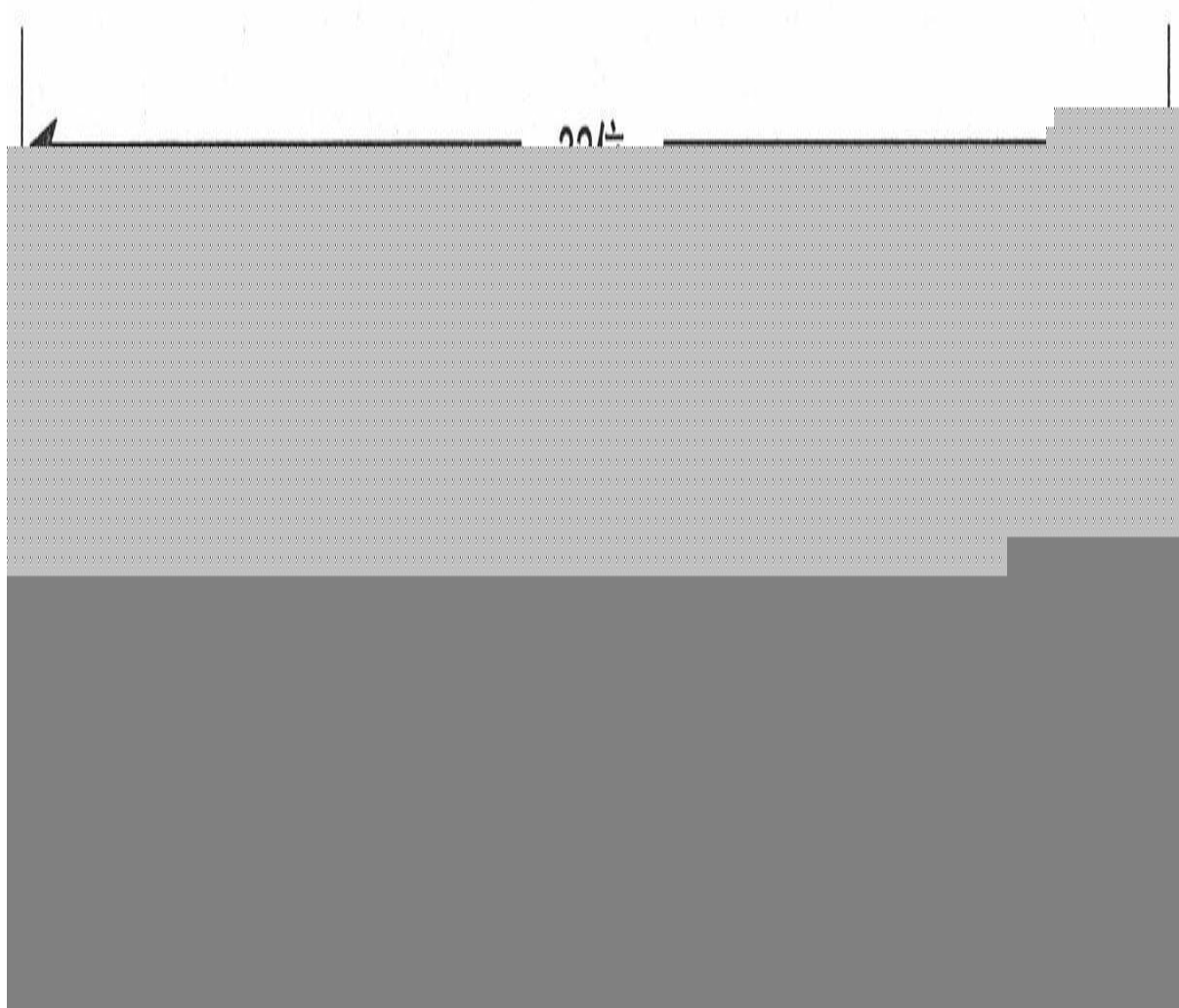
在图8-47中，路由器Dali的链路从路由器Matisse移到Goya的接口上，相关的IP地址也作相应的变动。路由器Goya现在成为了一台ASBR路由器，并把RIP学习到的路由重新分配到OSPF当中。

当一台ABR路由器同时也是一台ASBR路由器，并且和一个非纯末梢区域相连时，它的缺省行为是通告重新分配的路由到这个NSSA中去，参见示例8-53。



图8-47 路由器**Dali**的链路移到了路由器**Goya**上，现在变成路由器**Goya**来宣告**RIP**到路由器**Dali**，并且重新分配学到的路由到**OSPF**

示例8-53 一台**ABR**路由器同时也是一台**ASBR**路由器时，将会利用类型**7**的**LSA**通告外部路由到**NSSA**区域。在这个例子中，路由器**Goya**正在使用**N1**的度量类型通告外部路由



这个在ABR/ASBR路由器上缺省的路由重新分配行为可以通过在命令 **area nssa** 之后增加参数 **no-redistribution** 来关闭。这样，在所述例子的网络中，就不应该有类型3、类型4、类型5或类型7的LSA从ABR路由器发送到区域192.168.10.0。所需要的路由重新分配可以通过示例8-54的配置在路由器Goya上完成。 [\[29\]](#)

示例8-54 路由器Goya的配置不把RIP路由重新分配到NSSA 192.168.10.0

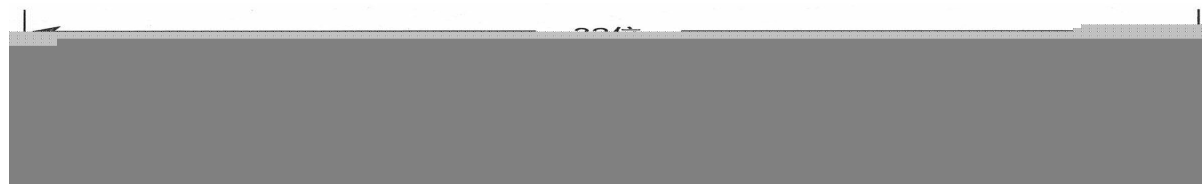
在这里，**area nssa** 命令阻塞了类型5的LSA通过路由器Goya进入到该区域，而**no-redistribution** 阻塞了类型7的LSA，命令**no-summary** 阻塞了类型3和类型4的LSA。和前面一样，命令**no-summary** 使路由器Goya发送一条类型3的LSA来向该区域通告一个缺省路由。在示例8-55中，显示了在路由器Goya禁止了类型7的路由重新分配之后路由器Matisse的路由表。注意，即使外部网络不在这个路由表中，它们依然是可达的，这是因为路由表中含有缺省路由的缘故。

示例8-55 在路由器Goya的**area nssa**命令中增加了**no-redistribution**参数后，示例8-53中的路由表将不再包括从类型7的LSA中学到的路由

在最后的这个例子中，假设需要路由器Goya允许类型3和类型4的LSA泛洪扩散到NSSA区域，但是不允许类型5和类型7的LSA泛洪扩散到该区域。这里的问题是当在路由器上去除了**no-summary** 参数后，ABR路由器将不再始发一条类型3的LSA通告缺省路由了。没有缺省路由，外部

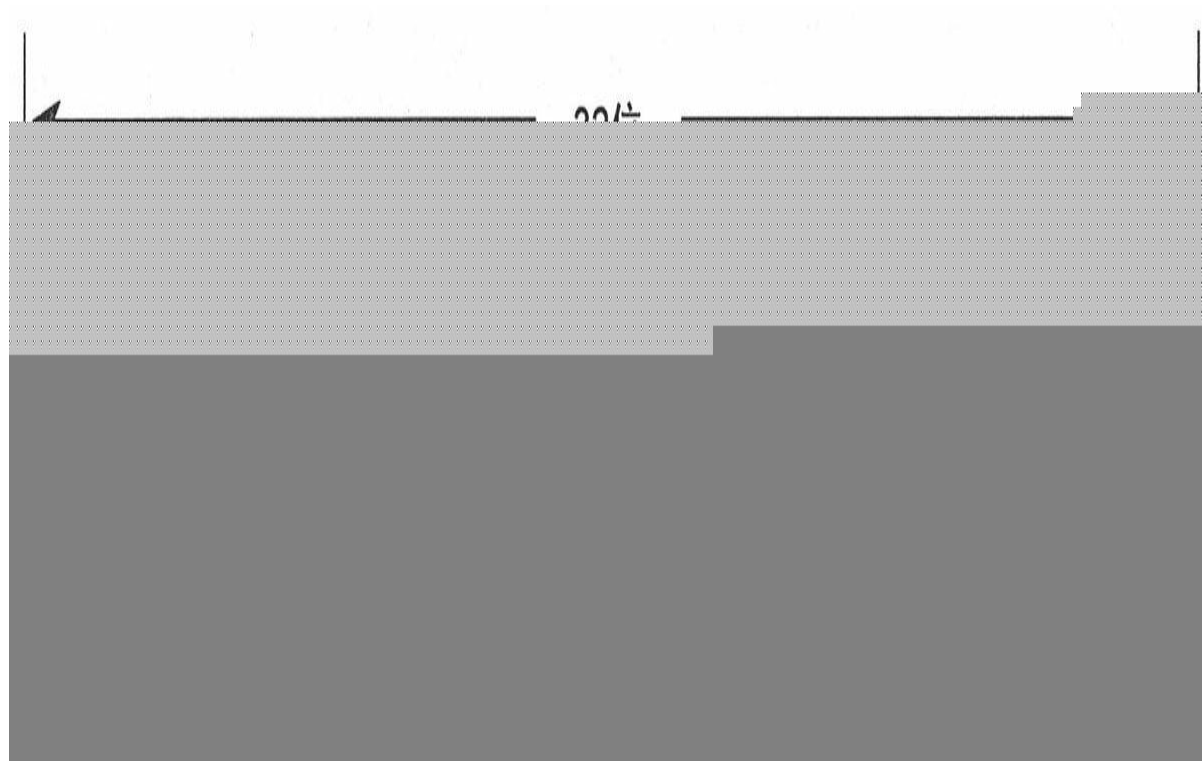
网络也就无法从NSSA区域内部到达。要解决这个问题，可以在命令 **area nssa** 后面增加一个 **default-information-originate** 参数，这就可以使用ABR路由器来通告一条缺省路由到这个NSSA区域。这时，它是使用类型7的LSA来通告这条缺省路由的。要使用这个参数，路由器Goya上的OSPF配置参见示例8-56。

示例8-56 路由器Goya使用**default-information-originate**命令发起缺省路由



在示例8-57中，显示了做过重新配置后的路由器Matisse的路由表。在这里，路由表中包括了区域间路由和一条缺省路由，这条缺省路由带有N2标志，表明这条路由是从类型7的LSA学习到的。

示例8-57 在命令**area nssa**中增加**default-information-originate**参数后，可以使ABR路由器通告一条缺省路由到这个NSSA区域



8.2.8 案例研究：地址汇总

虽然末梢区域可以通过防止某些LSA进入该区域，从而达到在一个非骨干的域中节省资源的目的，但是从骨干区域上来看，这些区域除了节省资源外并没有做其他任何事情。一个区域内所有的地址仍然会通告到骨干区域中。这种情形可以通过地址汇总来帮助解决。像末梢区域一样，地址汇总也是通过减少泛洪扩散的LSA数量来达到节省资源的目的。另外，它还可以通过屏蔽一些网络不稳定的细节来节省资源。例如，一个忽好忽坏的不稳定的子网在它每一次状态发生转变的时候，都会引起LSA在整个网络中进行泛洪扩散。但是，如果这个子网地址被汇总后包含在一个汇总地址中，那么单独的子网和它的稳定性就不再被通告出去了。

在Cisco路由器上可以执行两种类型的地址汇总：区域间路由汇总和外部路由汇总。区域间路由汇总（inter-area summarization），顾名思义，是指在区域之间的地址汇总。这种类型的汇总通常是配置在ABR路由器上的。外部路由汇总（external route summarization）允许一组外部地址汇总为一条汇总地址并通过重新分配注入到一个OSPF域中，这种类型的汇总通常是配置在ASBR路由器上的。区域间路由汇总将在本小节介绍，而外部路由汇总将在第11章讲述。

在图8-48中，区域15包含了8个子网：10.0.0.0/16~10.7.0.0/16。图8-49显示了这些子网地址可以使用单个汇总地址10.0.0.0/13来表示。

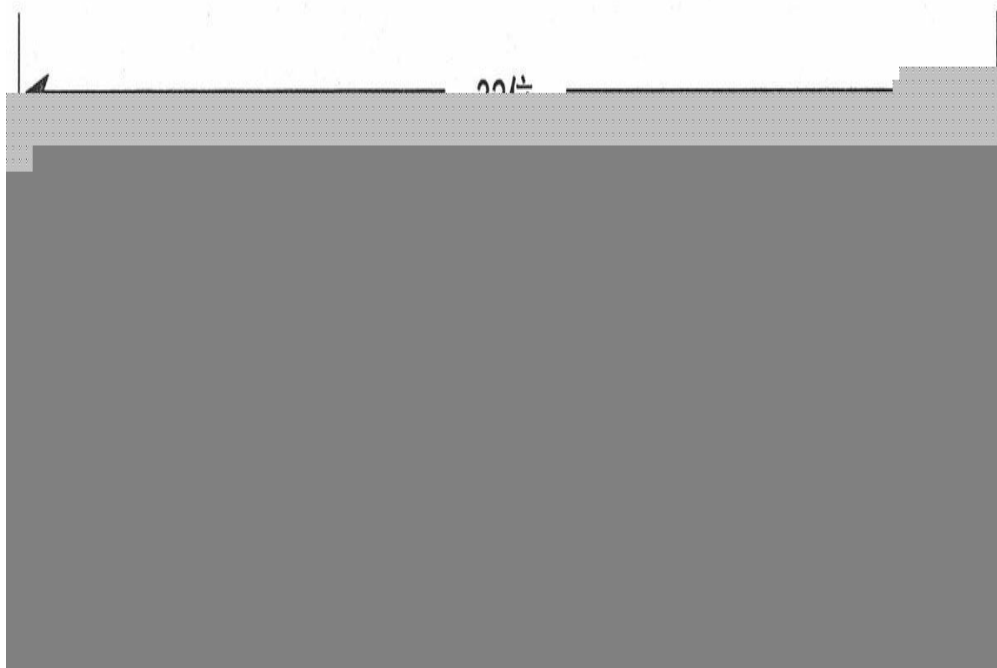


图8-48 在区域15和区域25中的地址能够汇总到骨干区域里

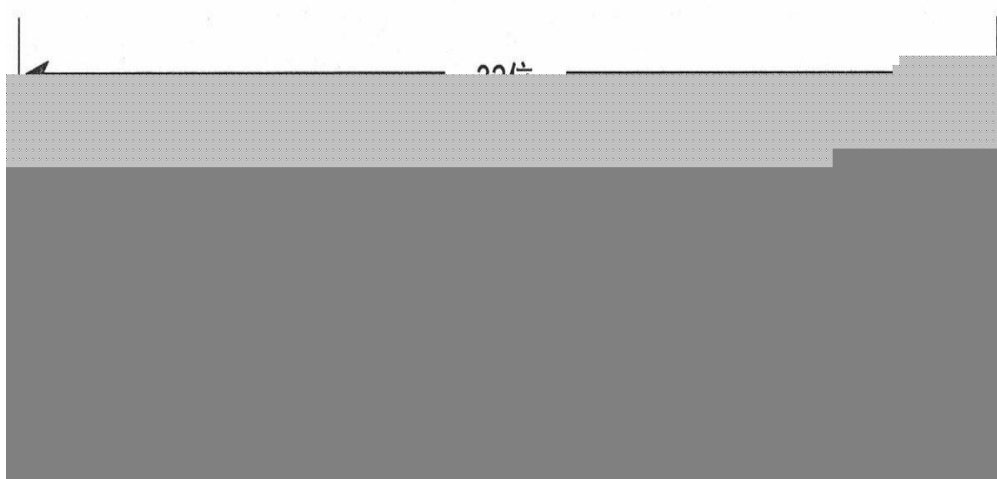


图8-49 可以使用汇总地址10.0.0.0/13来表示地址10.0.0.0/16~10.7.0.0/16的地址范围

在一台ABR路由器上配置一个汇总地址，既可以通告到骨干区域，也可以通告到一个非骨干的区域。最好的方法是，一个非骨干区域的地址应该通过它自己的ABR路由器汇总到骨干区域。而与之相对的是使其他所有的ABR路由器将这个区域汇总到它们各自的区域内。然后，在骨干区域中，被汇总的地址将穿越骨干区域并通告到其他区域中去。这两种方

法都简化了路由器的配置并减小了骨干区域内链路状态数据库的大小。

area range 命令指定了汇总地址所属的区域、汇总地址和地址掩码。回忆第7章的内容，当为EIGRP协议配置一条汇总路由时，会有一条指向空接口（**null Interface**）的路由被自动地加入路由表中，以便用来避免路由黑洞和路由环路。[\[30\]](#) OSPF也不会自动地加入这条路由。但是，在IOS软件早于12.1版的主要版本中，不会自动地加入**null**接口的路由。在能够创建这条路由的IOS软件版本中，路由器也可以使用命令**no discard-route**不在它的路由表中加载这条路由。因此，无论何时，在一个OSPF域内配置汇总路由时，都应该确认是否为这条汇总地址增加了指向**null**接口的静态路由。如果它没有被自动创建，那么必须增加这条路由 或使用**discard-route**命令。

示例8-58显示了路由器Pena的OSPF配置。

示例8-58 路由器Pena的OSPF配置

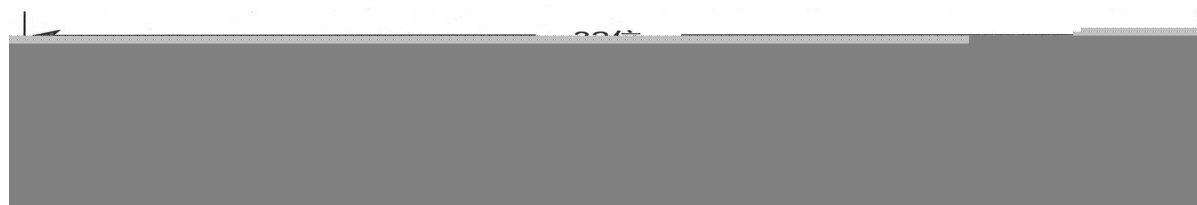
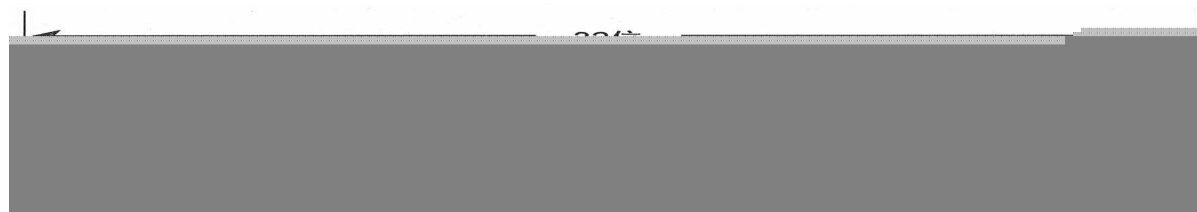


图8-49中显示出10.0.0.0/13表示的地址范围是连续的，也就是说，被汇总的3个二进制位构成了每一个000~111的组合。而在区域25中的地址不同，它们不能形成一个连续的地址范围。但是，它们仍然可以通过示例8-59中的配置在路由器Hurd上进行汇总。

示例8-59 路由器Hurd的OSPF配置



即使在这个汇总地址范围内的地址出现在这个网络以外，该汇总路由也是可以正常工作的。在图8-48中，网络172.17.0.0/16在区域15内，尽管这个地址是属于来自区域25汇总的那一段地址的。路由器Pena将通告这

个地址到骨干区域，而路由器Hurd将学到它并通告到区域25中去。跟随这个地址的网络掩码比汇总地址172.16.0.0/12的掩码更具体（也就是说，是更长）；又因为OSPF协议是无类别的路由选择协议，因此，OSPF将能够转发属于网络172.17.0.0的目的地址到正确的目的地。

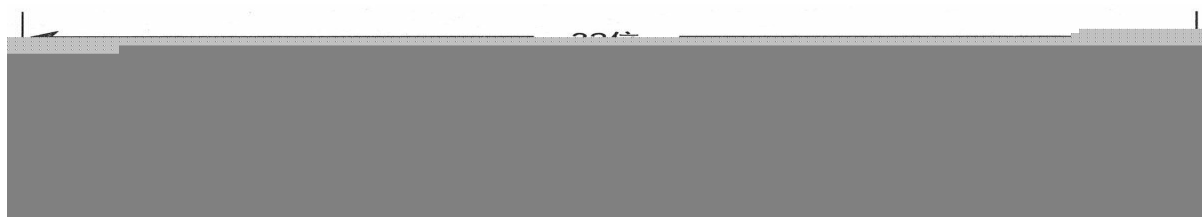
尽管图8-48中的地址配置方法可以工作，但是这并不是一个值得推荐的设计方法。汇总地址的地方是节省了资源，但对网络172.17.1.0/24的通告还是必须独立于网络172.16.0.0/12，并穿过骨干区域。在像这样的网络设计中，如果使用了缺省路由，还可能会产生路由环路的隐患。这个问题会在第12章中进行详细讲述。

这里也要注意图8-48中，子网172.16.27.0/25（和路由器Gorman相连）和子网172.16.27.192/29（和路由器Okeeffe相连）是不连续的。又因为OSPF协议是一个无类别路由选择协议，因而路由器Gorman和Okeeffe不会扮演网络边界路由器的角色。这些子网和它们的掩码将会通告到网络172.20.0.0中去，并且不会有路由选择不确定的情况发生。

area range 命令的缺省行为是通告指定的范围。它也可以用来抑制一个地址范围的通告。带有关键字**not-advertise** 的命令**area range** 将使指定范围的前缀被抑制。它们将不会在LSA中通告。

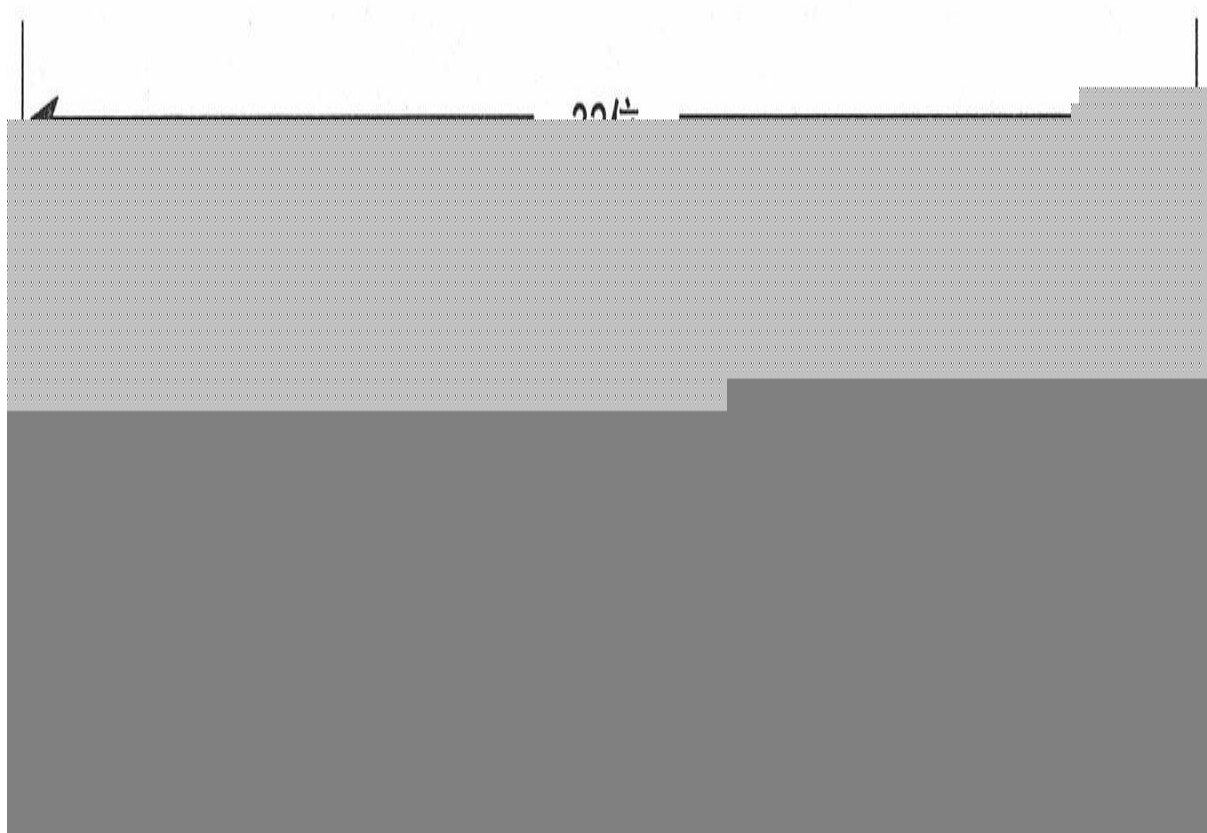
配置路由器Pena来抑制范围172.17.0.0/16。路由器Pena的配置参见示例8-60所示。

示例8-60 路由器**Pena**的配置抑制了指定地址范围的通告

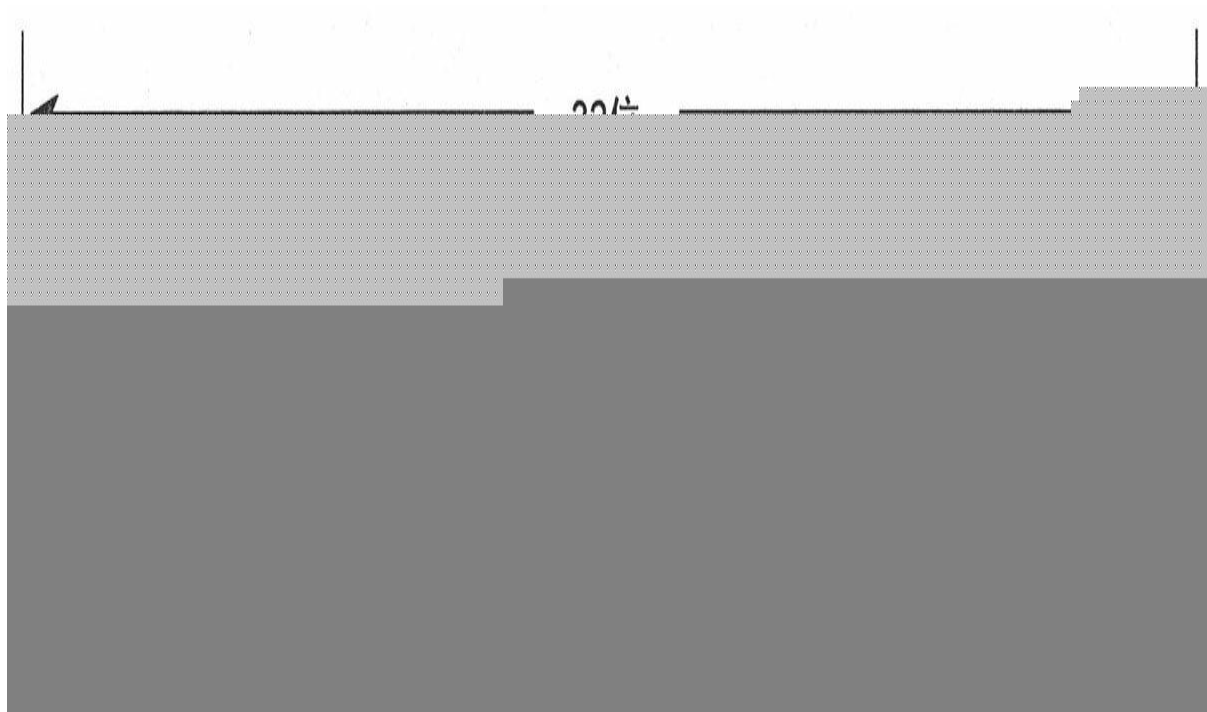


这条命令达到了抑制172.17.0.0/16地址范围的预期影响，但是也产生了一个不希望出现的结果。请查看下面路由器Hurd在配置这条命令之前（参见示例8-61）和之后（参见示例8-62）的路由表。

示例8-61 在路由器**Pena**抑制一个地址范围之前，路由器**Hurd**的路由表中显示了两条外部路由

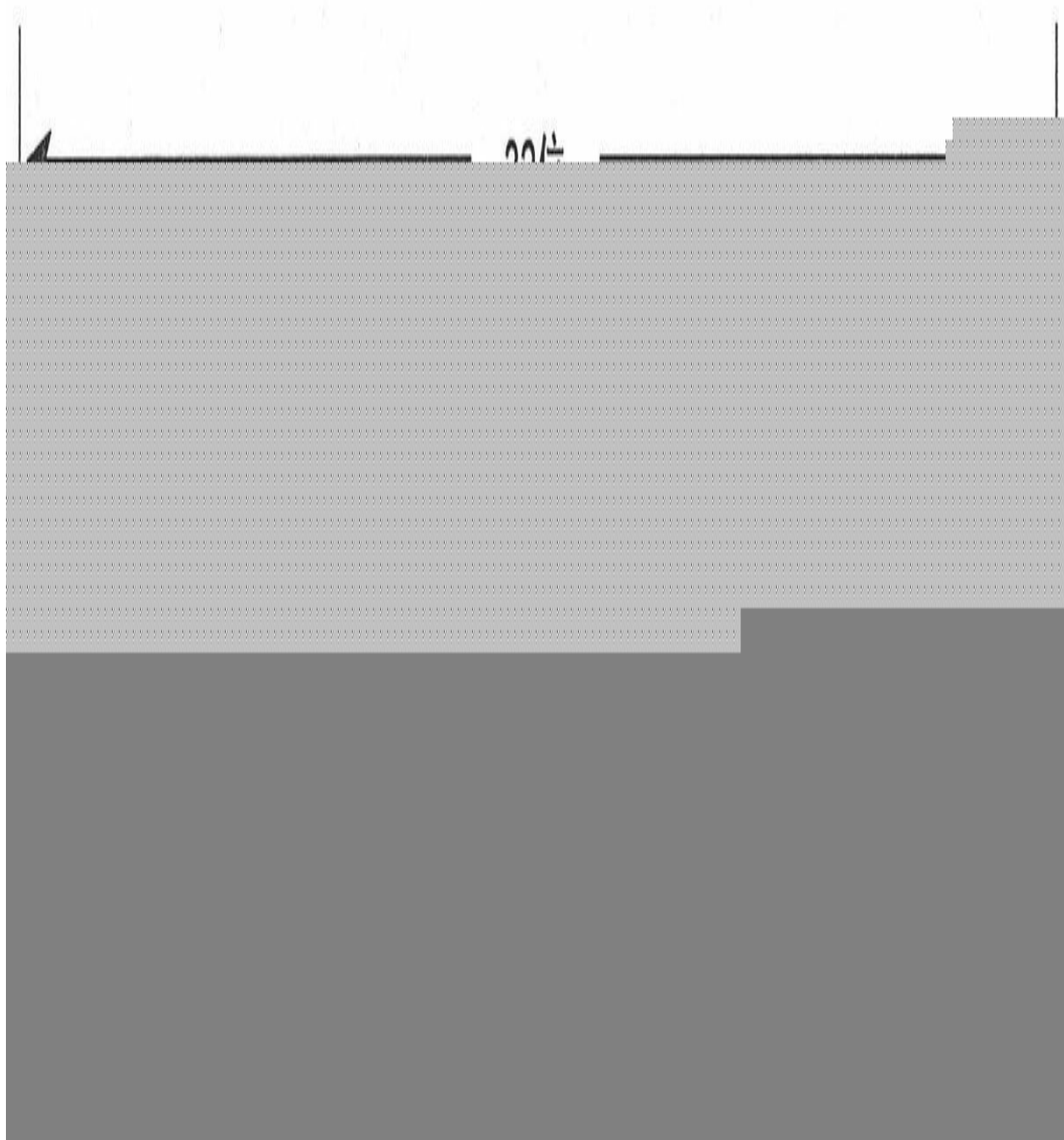


示例8-62 在路由器**Pena**抑制一个地址范围之后，路由器**Hurd**的路由表中显示这两条外部路由已经在路由表中了



在路由器Wyeth上通过RIP学习到的外部路由10.100.0.0和10.101.0.0已经不在路由器Hurd的路由表中了。示例8-63显示了Hurd的OSPF数据库中的10.100.0.0。请注意指定的转发地址。

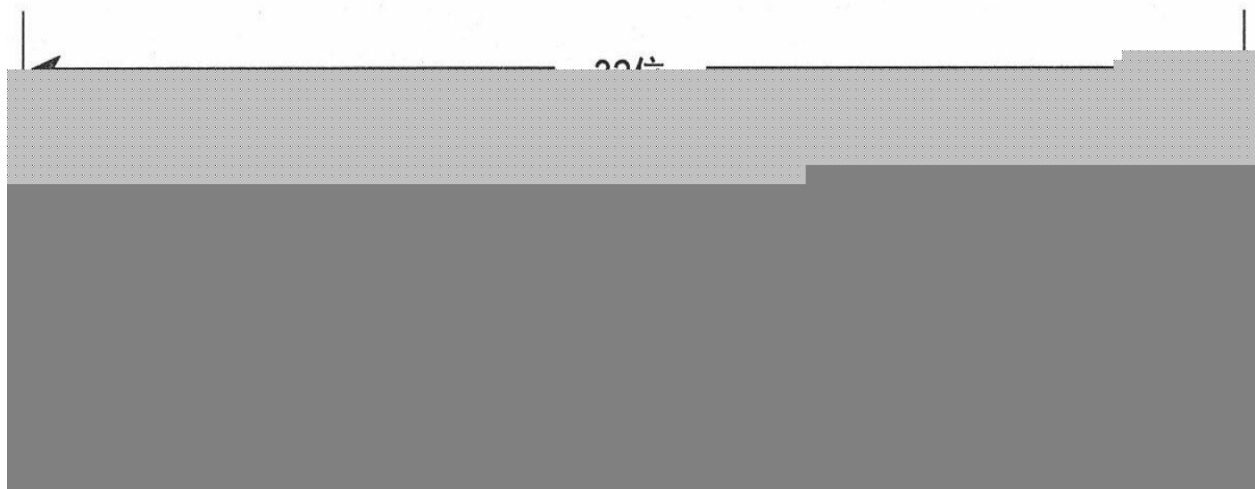
示例8-63 类型5的外部链路状态显示转发地址是始发这条外部路由的路由器的地址



这里的转发地址是路由器Wyeth的loopback接口。如果指定了外部LSA的转发地址，而这个地址是不可达的，那么这个LSA中包含的地址就不会插入到路由表中。当路由器Pena将类型7的NSSA LSA转换为类型5的LSA时，缺省情况下转发地址将从类型7转换为类型5。在转换期间，可以配置ABR路由器来抑制它的转发地址，这可以使用地址0.0.0.0代替指定的地址。当另一台路由器接收这个抑制了转发地址的类型5外部LSA时，接收路由器将会把要到达外部地址的流量引导到转换类型7到类型5的ABR路由器上，而不是引导到转发地址。

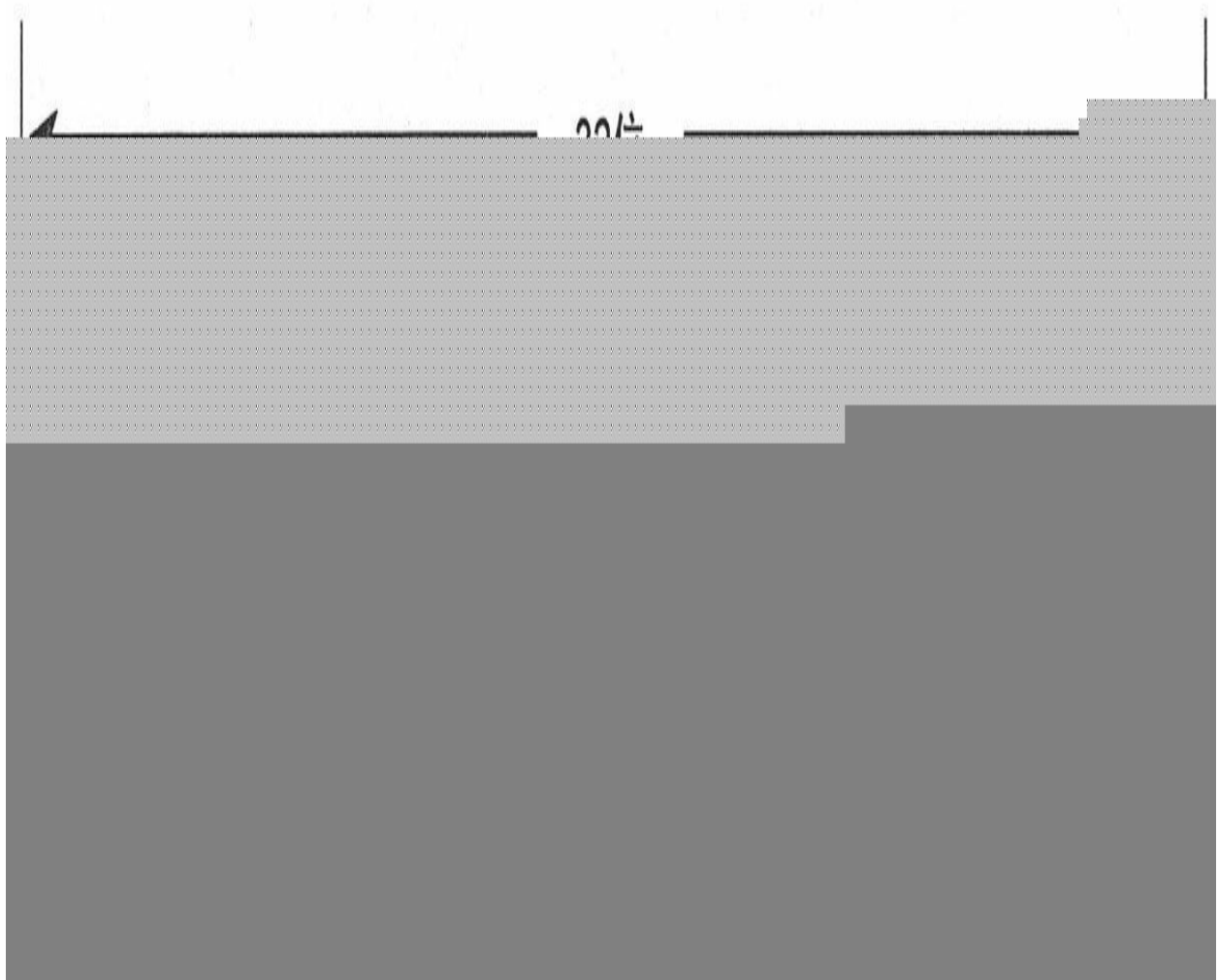
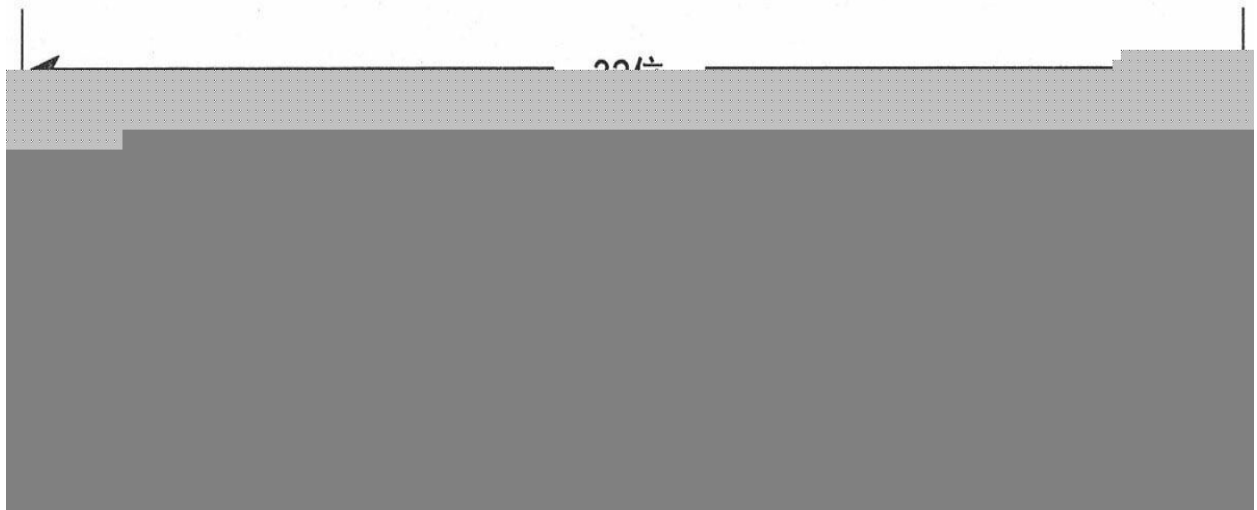
示例8-64显示了路由器Pena的新配置。

示例8-64 路由器Pena的配置抑制了包括在转换类型5 LSA中的转发地址

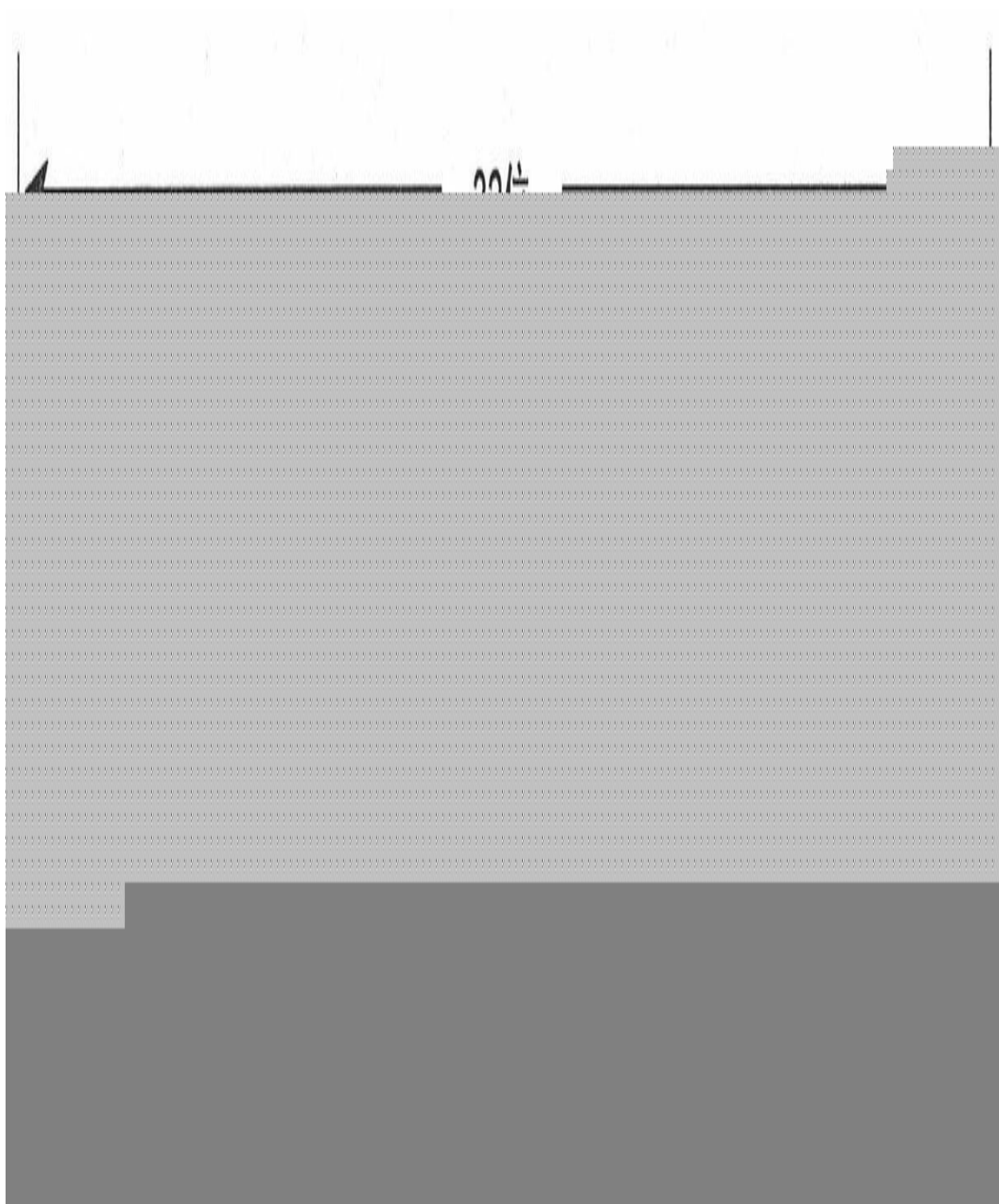


示例8-65显示了路由器Hurd的外部数据库中有关10.100.0.0的条目，示例8-66显示了路由器Hurd的新路由表。

示例8-65 外部OSPF数据库的条目显示了一个被抑制的转发地址



示例8-66 在抑制了转发地址后，外部路由**10.100.0.0**和**10.101.0.0**被插入到路由表中



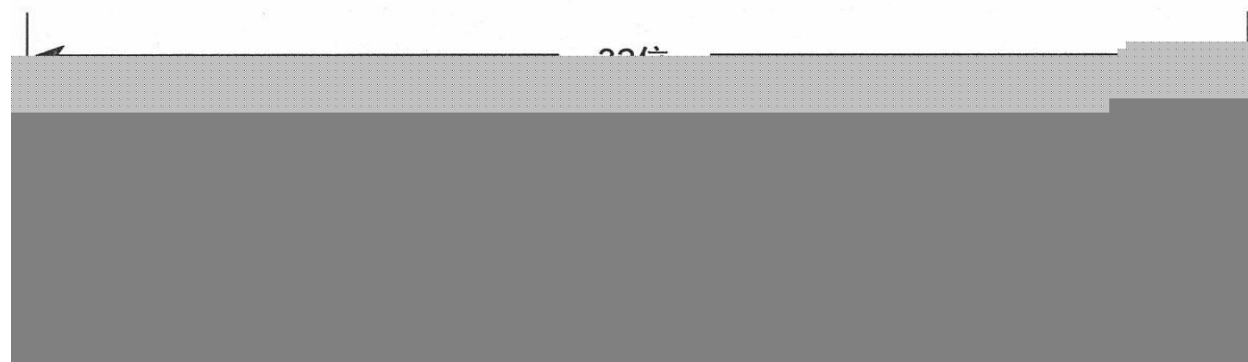
在路由器Hurd的数据库中显示的转发地址现在是0.0.0.0，原来两条外部路由回到了路由表中。

8.2.9 案例研究：在区域之间进行过滤

在路由器Okeeffe上增加另外一个LAN。LAN上的设备可以被区域25内的其他设备访问，但不能被这个网络中其余的设备访问。在区域之间限制和控制地址交换的另一种方法是在ABR上配置类型3的LSA过滤。ABR可以过滤由类型3的LSA通告的网络地址进出某个区域。

在区域25的路由器Okeeffe增加LAN的地址192.168.1.0/24。这个在LSA中通告的地址虽然只能被区域25内的设备访问，但可以通过整个网络。为了避免这个地址被通告到区域25的外部，可以在路由器Hurd上配置类型3的LSA的过滤，参见示例8-67。

示例8-67 路由器Hurd应用了类型3的LSA的过滤



OSPF命令**area *PID* filter-list prefix** 指定了应用于入站或出站LSA的过滤列表的名称。出站列表过滤了发送到由该命令指定的区域之外的其他区域的LSA。在我们的例子中，该列表过滤了由区域25发起的，并发送到非区域25的区域的地址的LSA，例如区域0。入站列表过滤了发送到区域25的LSA。

前缀列表第一行的作用是清楚的。这条语句拒绝地址192.168.1.0/24由类型3的LSA通告出去。第二行语句允许所有其他地址通过：从掩码长度为0位到32位的所有地址。这里第二行的语句是必须的，因为在前缀列表的末尾缺省隐含的语句是“deny all”。地址192.168.1.0/24防止在类型3的LSA中发送到区域25外部，而其他所有的地址都被允许通过。

8.2.10 案例研究：认证

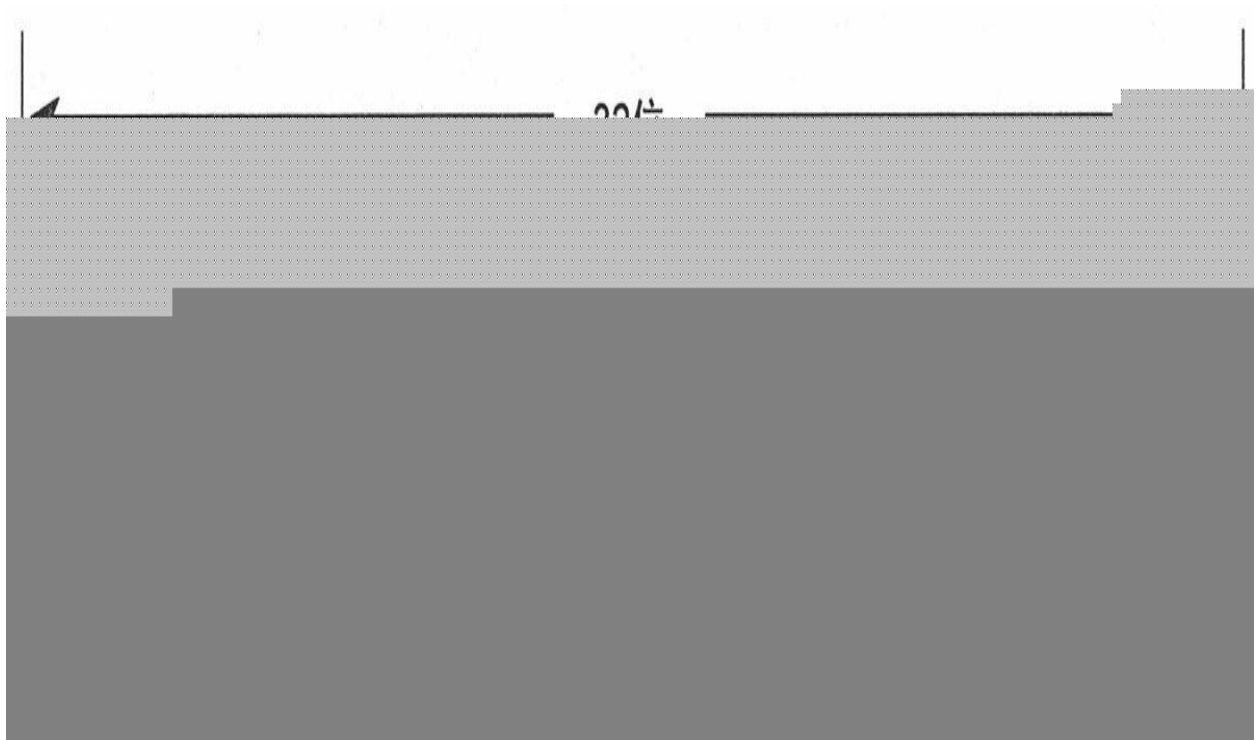
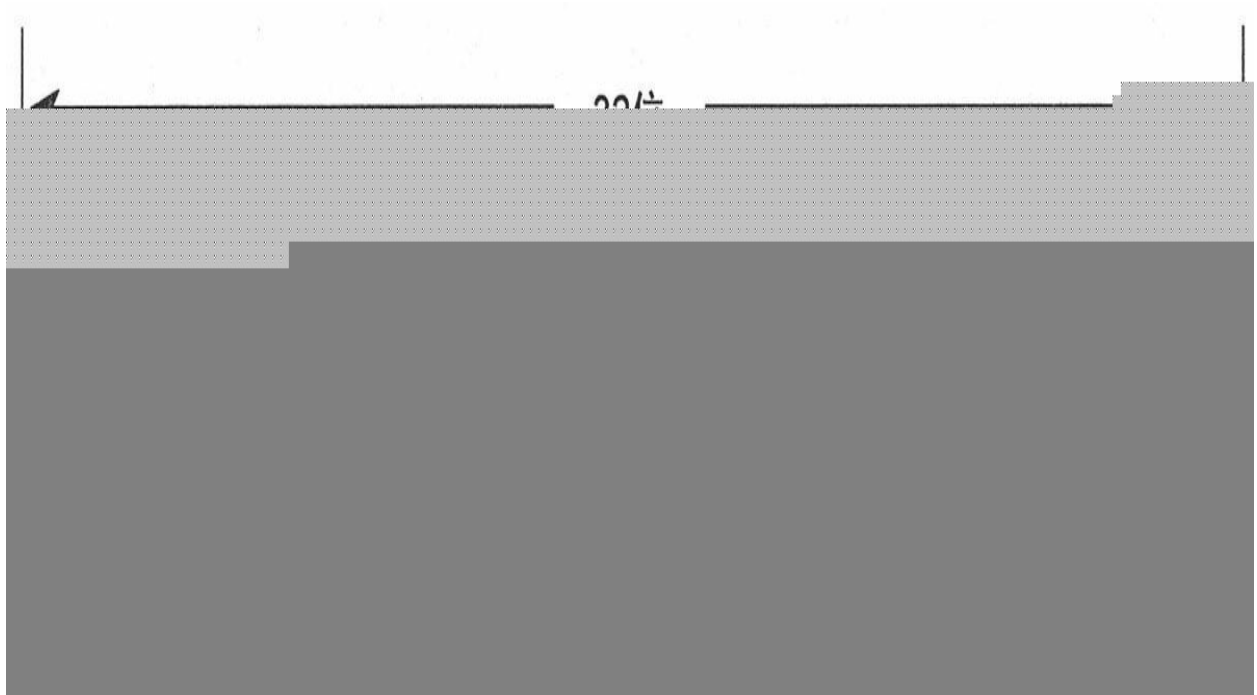
OSPF数据包可以通过认证来防止有意或无意地引入有害路由信息的情况发生。表8-8中列出了有效的认证类型。Null认证（类型0）是路由器缺省使用的类型，表示在数据包头部没有包含认证信息，也就是说，不需要认证。在路由器上可以使用简单的明文口令（类型1）或者MD5加密校验和（类型2）来配置认证。如果一个区域内某处配置了认证，那么就必须在整个区域内都配置认证。

如果网络管理者是以增加网络的安全性为目标的，那么只有在OSPF区域内的设备不能支持更安全的类型2的认证时才考虑使用类型1的认证。明文认证会在网络上给网络攻击者留下一个安全漏洞，因为网络上传送的数据包能够被协议分析仪捕获，并读出所设置的口令(请参考第6章，并注意图6-8)。但是，类型1的认证在进行OSPF的重新配置时会变得比较有用。例如，不同的口令可以用在进行重新配置时“旧”OSPF路由器和“新”OSPF路由器上，从而避免它们在共享一个公共广播网络的情况下相互通信。

在一个区域上配置类型1的认证方式，可以使用命令**ip ospf authentication-key** 来为和该区域相连的每一个接口分配一个口令，这个口令最长为8个八位组字节长。所分配的口令不必在整个区域上都相同，但是在一对邻居路由器之间必须相同。在OSPF协议的配置下添加**area authentication** 命令来使类型1的认证方式有效。

参考图8-48，在区域0和区域25上启用类型1的认证。路由器Hurd的配置参见示例8-68。

示例8-68 路由器**Hurd**配置了明文认证



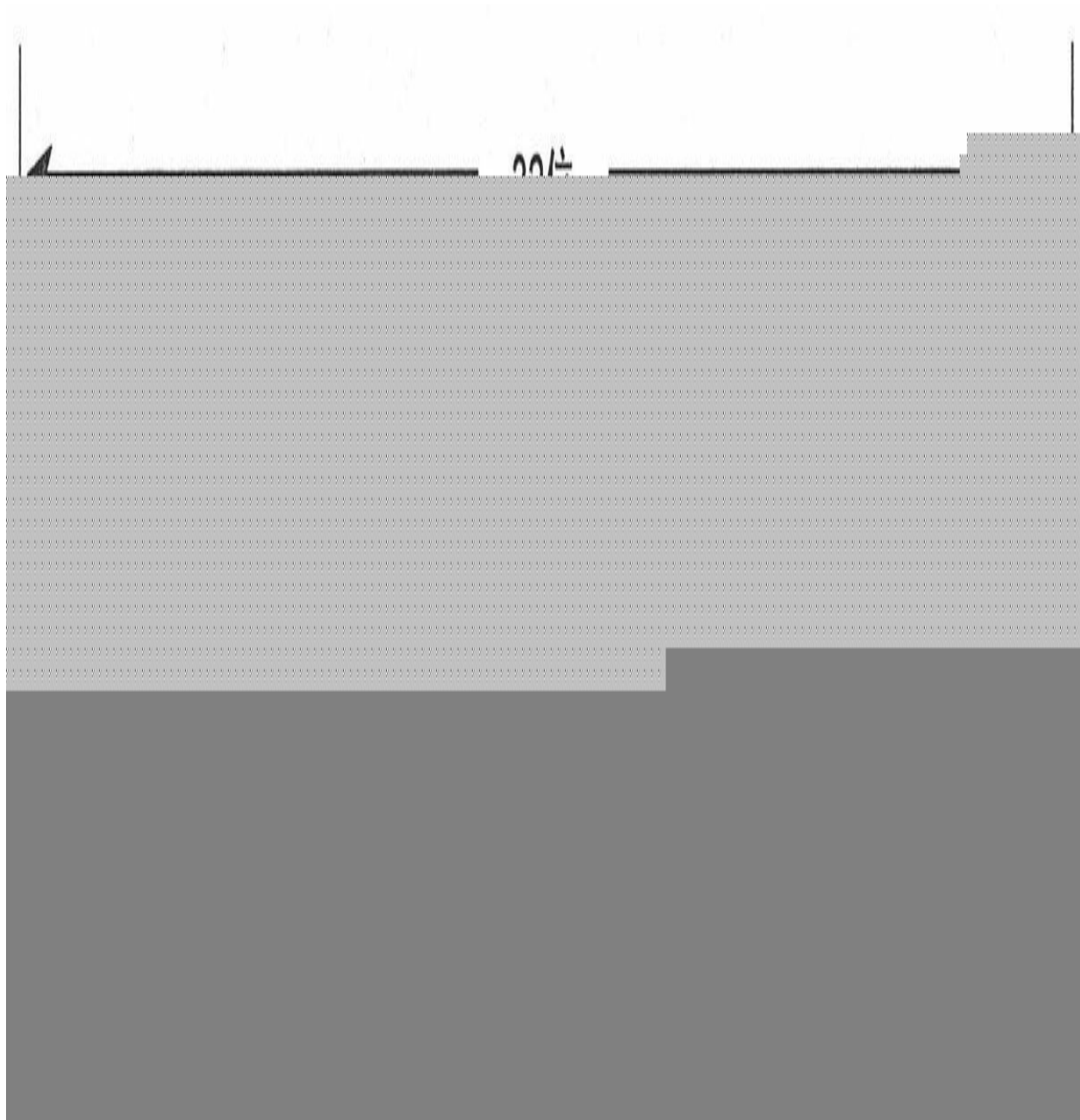
在这里，口令“santafe”用在路由器Hurd和Pena之间；口令“taos”用在路由器Hurd和Gorman之间；而口令“abiquiu”用在路由器Hurd和Okeeffe之间。

MD5算法用在类型2的认证方式中。它用来为OSPF数据包内容和一个口令（或密钥）计算一个散列值(hash value)。这个散列值将和一个密钥ID，以及一个不变小的序列号一起在数据包中传送。拥有相同口令的接收路由器将会计算它自己的散列值。如果传送的消息中什么内容都没改变，那么接收路由器的散列值应该和发送路由器在消息数据包中传送的散列值相匹配。密钥ID允许路由器指定多个口令，这样可以使口令的改变比较容易，并具有更好的安全性。在这个案例研究中包含了一个口令迁移的例子。序列号用来防止“重现攻击(replay attacks)”，以避免OSPF数据包被捕获、更改和重新传送给一台路由器。

在一个区域上配置类型2的认证方式，可以使用命令**ip ospf message-digest-key md5** 来为和该区域相连的每一个接口分配一个口令和一个密钥ID(Key ID)，这里的口令最长为16个八位组字节，而密钥ID在1~255之间。和类型1的认证方式相同，所分配的口令不必在整个区域上都相同，但是在一对邻居路由器之间的密钥ID和口令都必须相同。在OSPF协议的配置下添加**area authentication message-digest**命令来使类型2的认证方式有效。

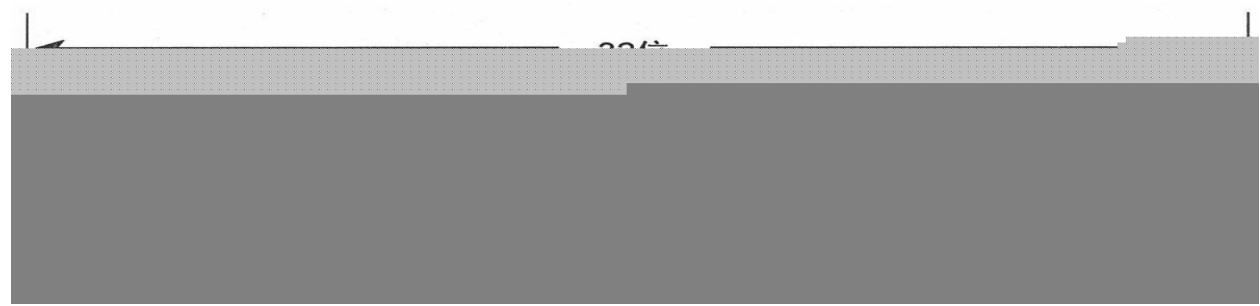
在路由器Hurd上启用类型2的认证方式，有关配置见示例8-69。

示例8-69 路由器Hurd配置了MD5认证



密钥允许路由器在不需要使认证无效的情况下更改它的口令。例如，为了在路由器Hurd和Okeeffe之间更改口令，新的口令可以和一个不同的密钥一起配置。示例8-70显示了路由器Hurd的配置。

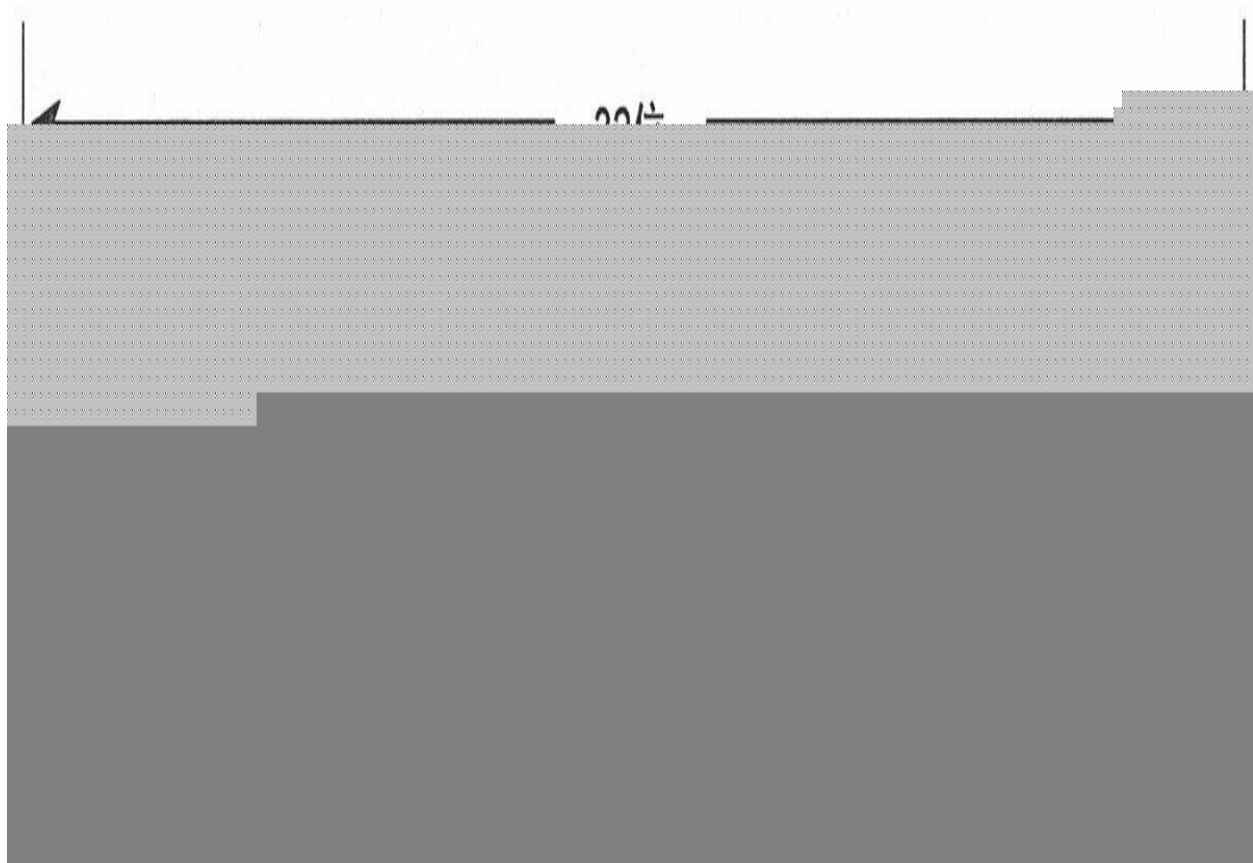
示例8-70 路由器**Hurd**配置了一个新的MD5钥匙



路由器Hurd现在将会在S0/0.2接口为每一个OSPF数据包发送两个重复的拷贝，一个使用密钥15认证，另一个使用密钥20认证。当路由器Hurd开始从路由器Okeeffe那里收到使用密钥20认证的OSPF数据包时，它就会停止发送密钥为15的认证数据包。一旦新的密钥可以使用了，原来的密钥就可以使用命令**no ip ospf message-digest-key 15 md5 abiquiu** 从这两台路由器上移掉。

一个在运行的网络中的口令从来不应该向这些例子中的口令那样可以预先得知。在所有使用认证的路由器的配置文件里增加命令**service password-encryption** 是比较明智的选择。这样做可以使路由器对配置文件中任何显示的口令进行加密，因此可以避免口令不被别人通过简单地查看路由器配置的文本拷贝就可以得知的隐患。如果已经给路由器Hurd的配置口令加密，那么，它的配置参见示例8-71。

示例8-71 在路由器**Hurd**上配置了口令加密，用来加密配置文件中显示的口令



在本书讲述的其他协议的认证配置中，钥匙链用于配置口令，或称为钥匙串。在编写本书的时候，OSPF不支持钥匙链的配置。

8.2.11 案例研究：虚链路

如图8-50所示，显示了一个骨干区域设计的比较糟糕的网络。如果路由器Hokusai和Hiroshige之间的链路失效了，那么这个网络的骨干区域将被分割成两部分。结果是，路由器Sesshiu和Okyo不能相互通信。即使这两台路由器是分离区域的ABR，区域间的通信量也将会在这些区域之间被阻塞。

在这个实例中，最有效的解决办法是在路由器Sesshiu和Okyo之间为骨干区域增加另外一条链路。在这个骨干区域得到改进之前，作为一种过渡方案，可以在路由器Hokusai和Hiroshige之间建立一条穿过区域100的虚链路。

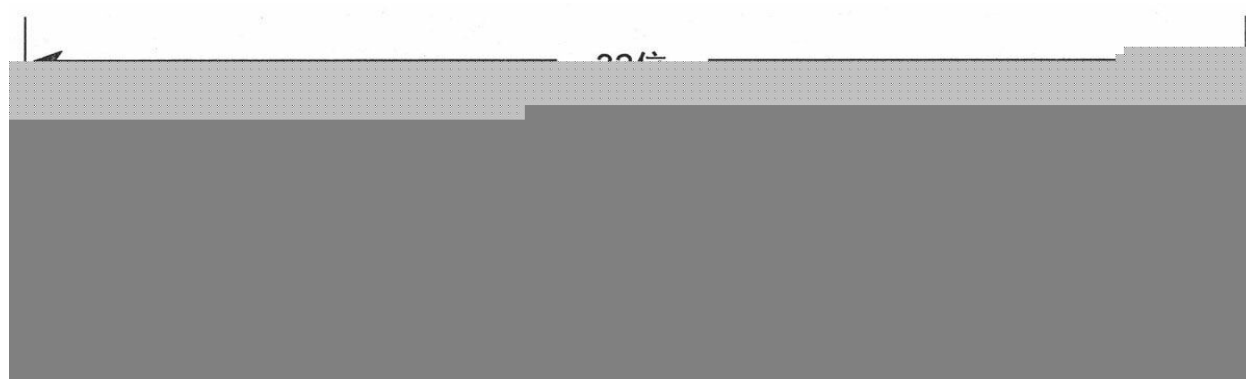
虚链路总是建立在ABR路由器之间的，至少它们之中有一台ABR路由器

是必须和区域0相连的。[\[31\]](#)在每一台ABR路由器的OSPF配置中，通过添加**area virtual-link** 命令来配置一条虚链路，并指定了这条虚链路要穿过的区域以及这条链路远端的ABR的路由器ID。在路由器Hokusai和Hiroshige之间建立一条虚链路的配置参见示例8-72和示例8-73。



图8-50 路由器Hokusai和Hiroshige之间的链路失效了将会使它们的骨干区域变成分段区域

示例8-72 路由器Hokusai的虚链路配置



示例8-73 路由器Hiroshige的虚链路配置

完成以上的配置后，在正常情况下，路由器Sesshiu和Okyo之间的数据包可以通过路由器Hokusai和Hiroshige之间的骨干区域上的链路进行转发。但是，如果那条链路失效，将会利用虚链路进行数据包的转发。在示例8-74中，虽然每一台路由器都把这条链路看作是一条无编号的点到点网络，但实际上数据包的转发是通过路由器Kujomoto的。[\[32\]](#)

示例8-74 使用命令**show ip ospf virtual-link**可以查看一条虚链路的状态

[8.2.12 案例研究：运行在NBMA网络上的OSPF](#)

在一些非广播多路访问(multi-access)网络上，例如X.25、帧中继和ATM

等，运行OSPF协议会产生一个问题。“多路访问”意味着一个NBMA的网络“云”是多台设备共同相连的单个网络，和以太网或者令牌环网络一样（如图8-51所示）。但是它又和以太网、令牌环网等广播型网络不同，它是非广播的。“非广播”意味着发送到这个网络上的数据包不一定会被和该网络相连的其他所有的路由器看到。这样，由于NBMA网络是多路访问的，OSPF协议将需要选取一台DR路由器和BDR路由器。但是由于NBMA网络又是非广播的，它不能保证所有相连的路由器都能收到其他所有路由器发送的Hello数据包。因此，所有的路由器不一定能够自动地了解它的所有邻居，因而不一定能够正确地进行DR的选取。

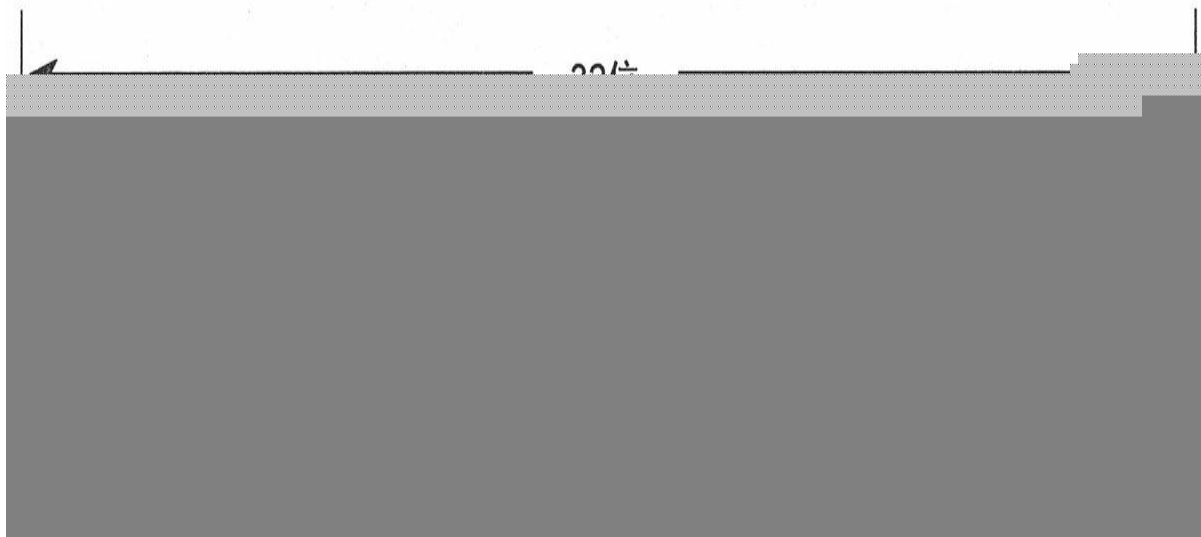


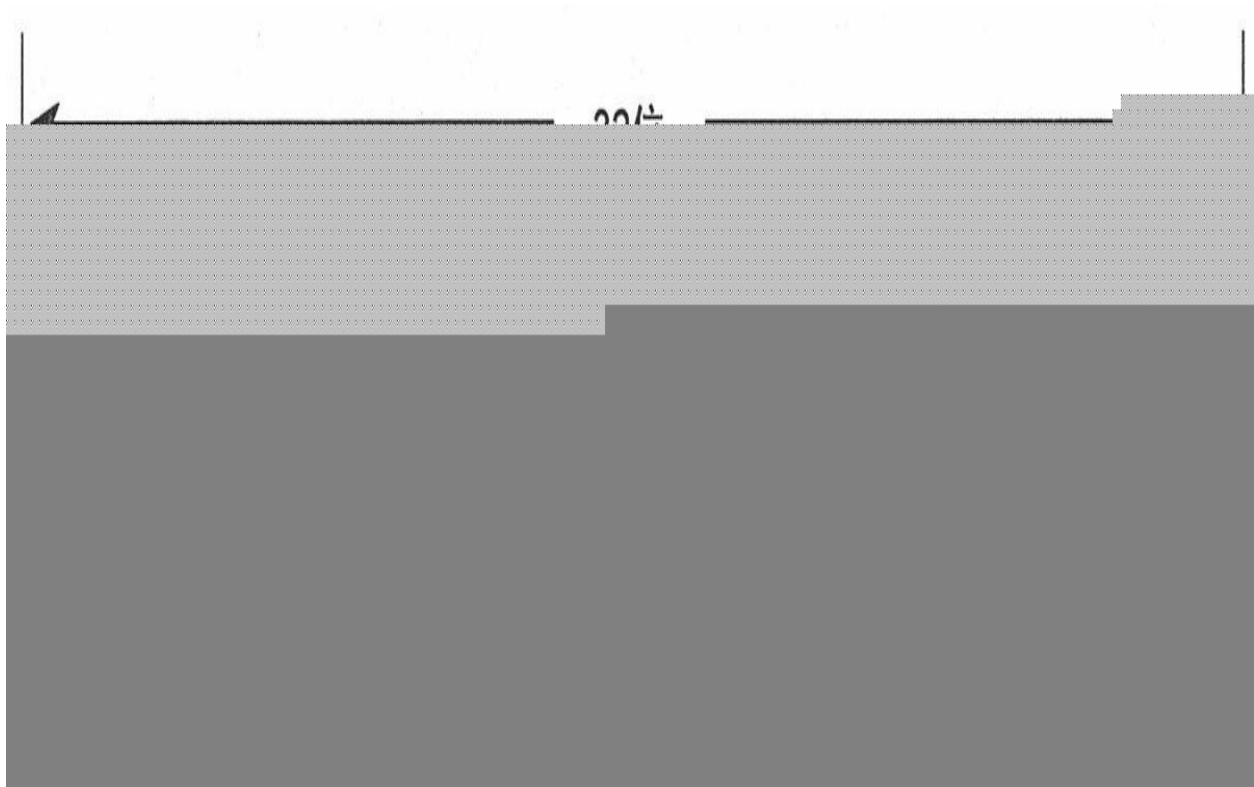
图8-51 路由选择协议会把NBMA网络看作是有多台设备相连的一个子网。但是当一个NBMA网络是部分网状连接时，正如这里所示，不是所有相连的路由器都和其他所有路由器直接连接

本小节内容将阐述几个解决NBMA网络问题的方案。具体方案的选择依赖于实现该解决方案的网络的特征。

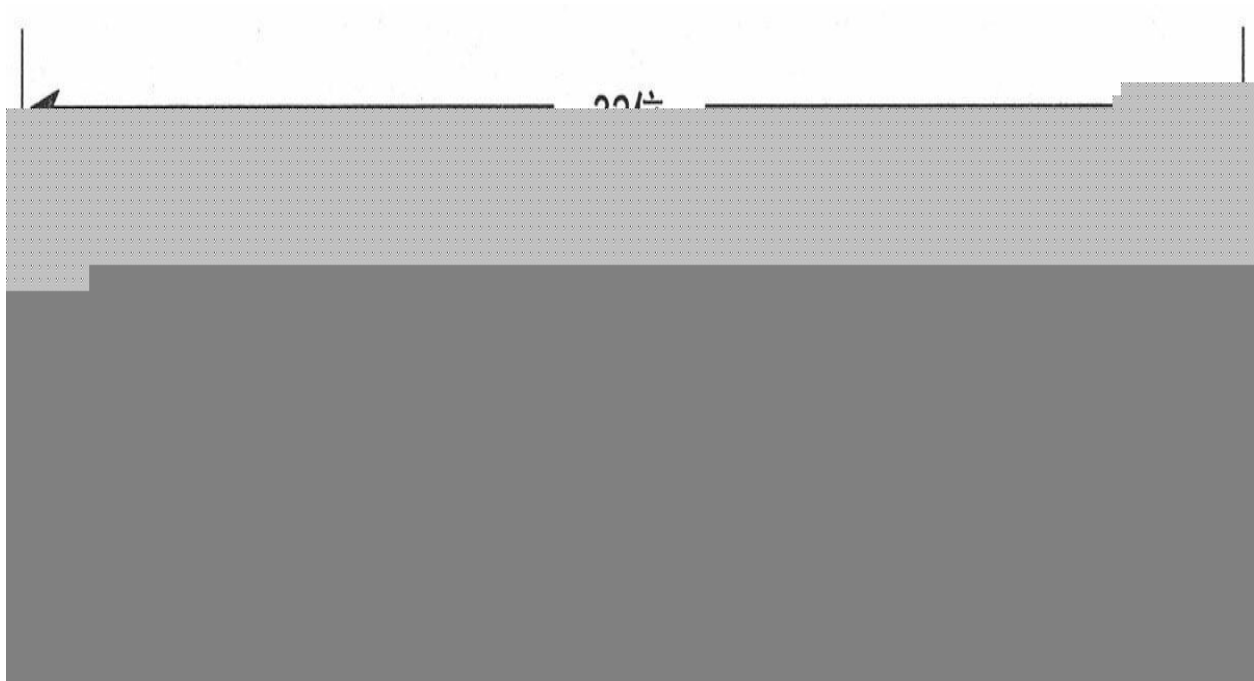
最早的解决方案出现在Cisco IOS 10.0版之前的有关版本中。它是使用neighbor命令以手工方式指定每一台路由器的邻居并创建DR的。如图8-52所显示了一个与4台路由器相连的帧中继网络。

在图8-52中，由于PVC电路的配置是部分网状连接的星型(hub-and-spoke)结构，因此，路由器Rembrandt必须成为一台DR路由器。作为中心(hub)，它是惟一的和其他所有的路由器直接相连的路由器。这4台路由器的配置参见示例8-75～示例8-78。

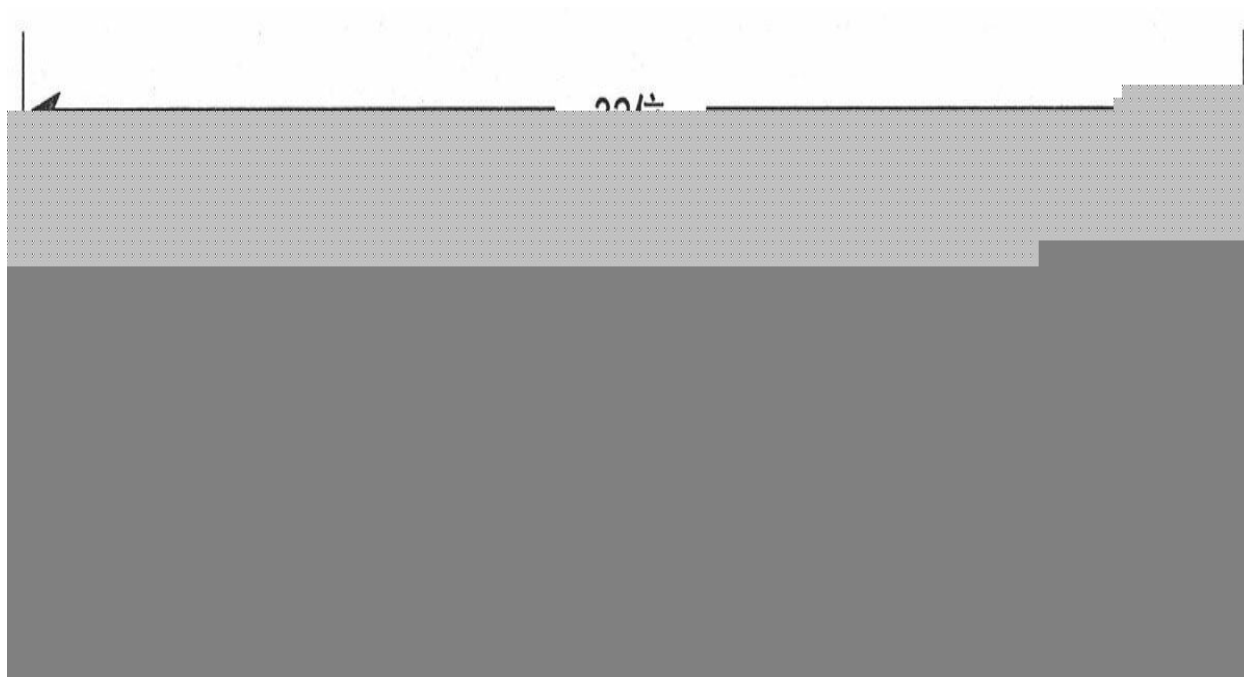
示例8-75 路由器Rembrandt的配置



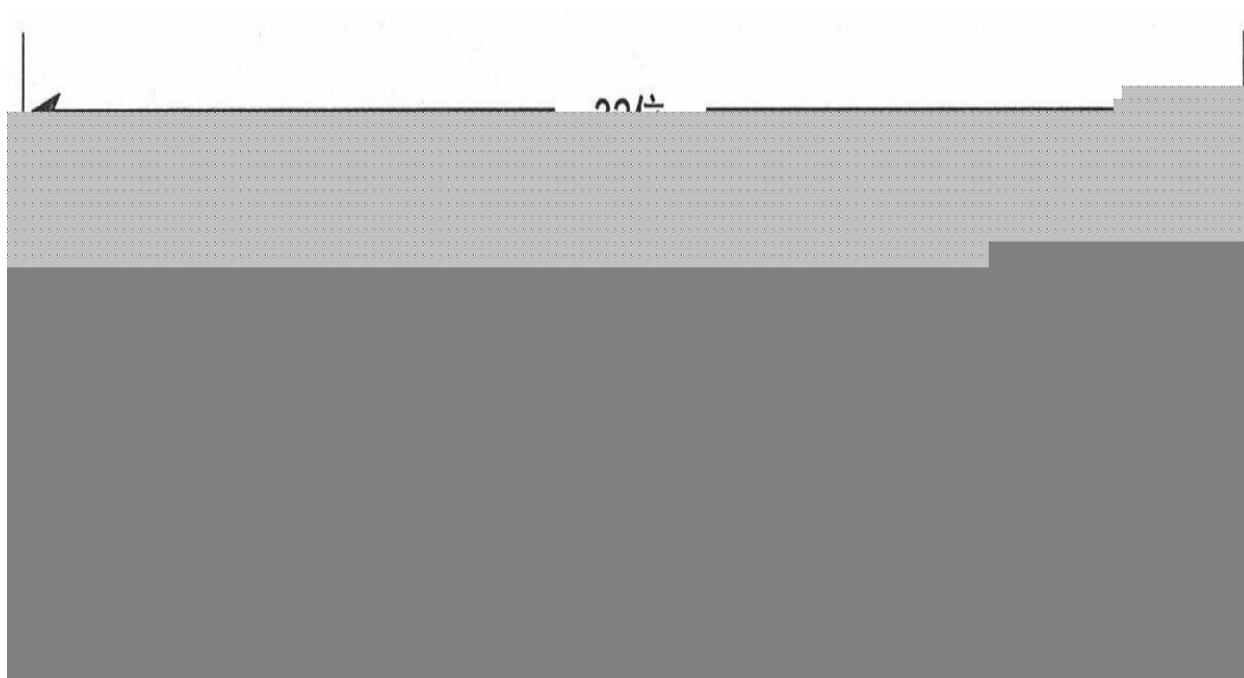
示例8-76 路由器Hals的配置指定了一个邻居的优先级



示例8-77 路由器Vandyck的配置指定了一个邻居的优先级



示例8-78 路由器Brueghel的配置指定了一个邻居的优先级



在路由器Rembrandt上，使用neighbor命令配置了它的3个邻居的接口IP地址。缺省的优先级是0，在路由器Rembrandt上不改变这个缺省值，没

有邻居有资格成为一台DR或者BDR。

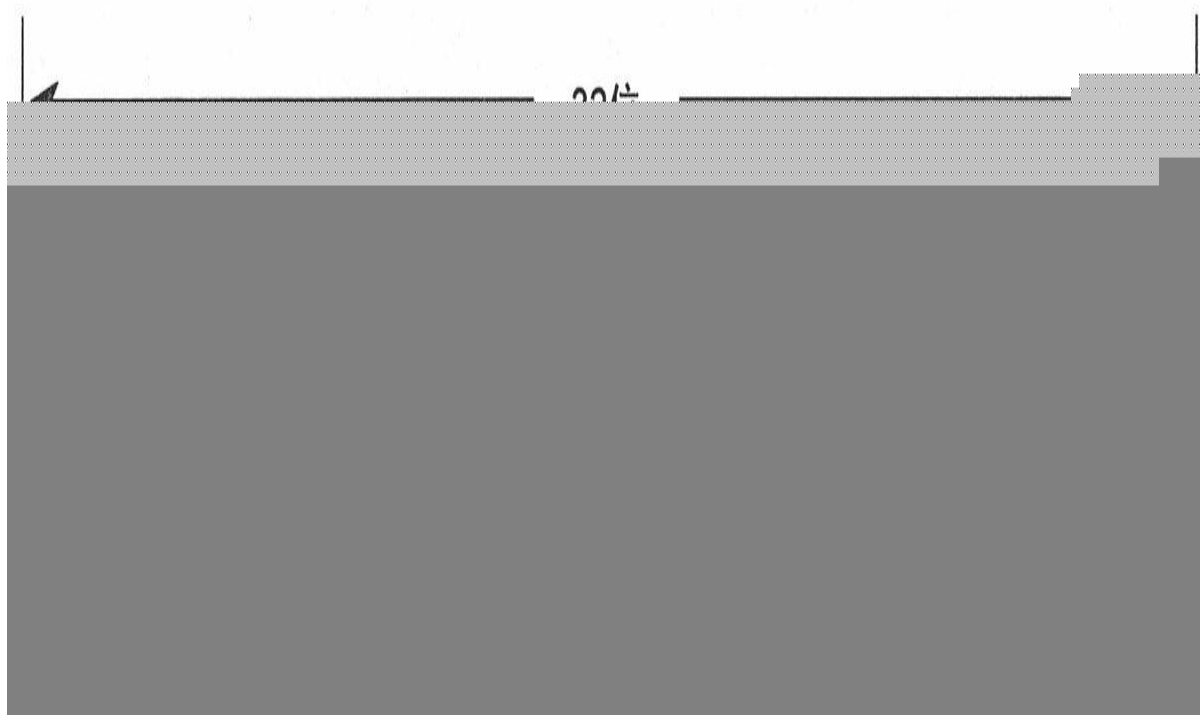
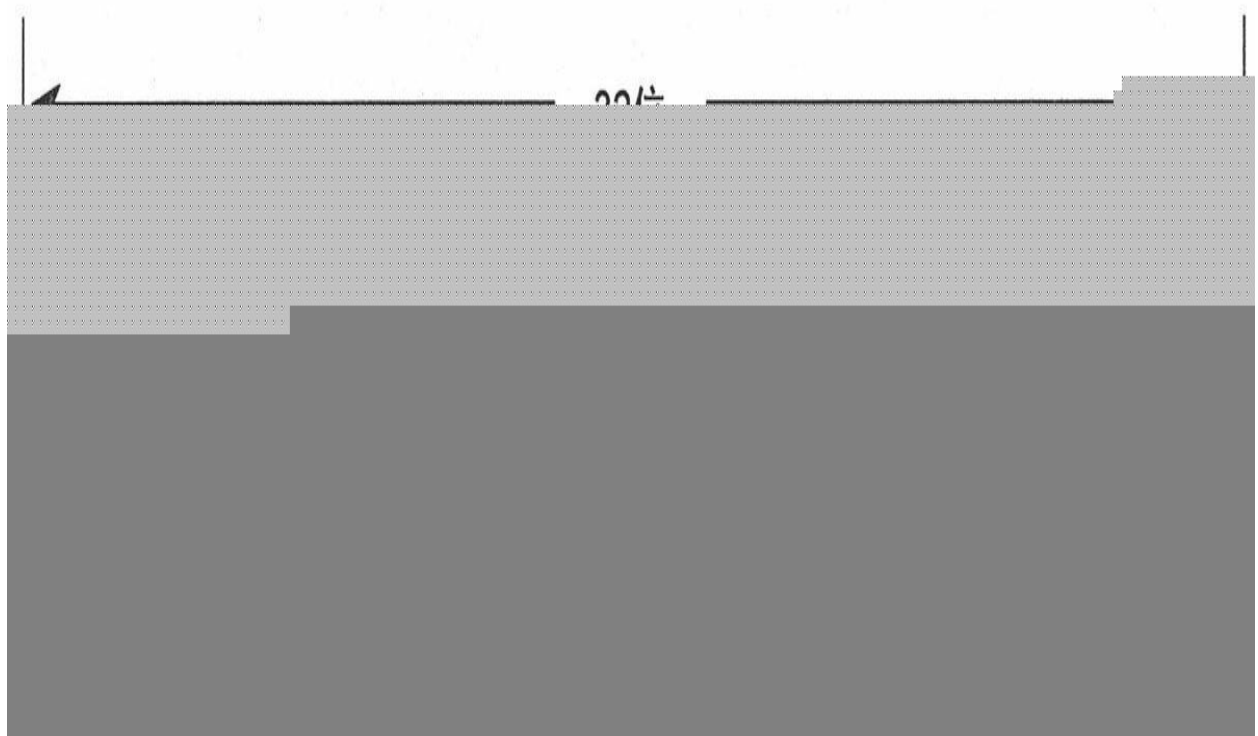


图8-52 在这个NBMA网络中配置OSPF存在的几种选择

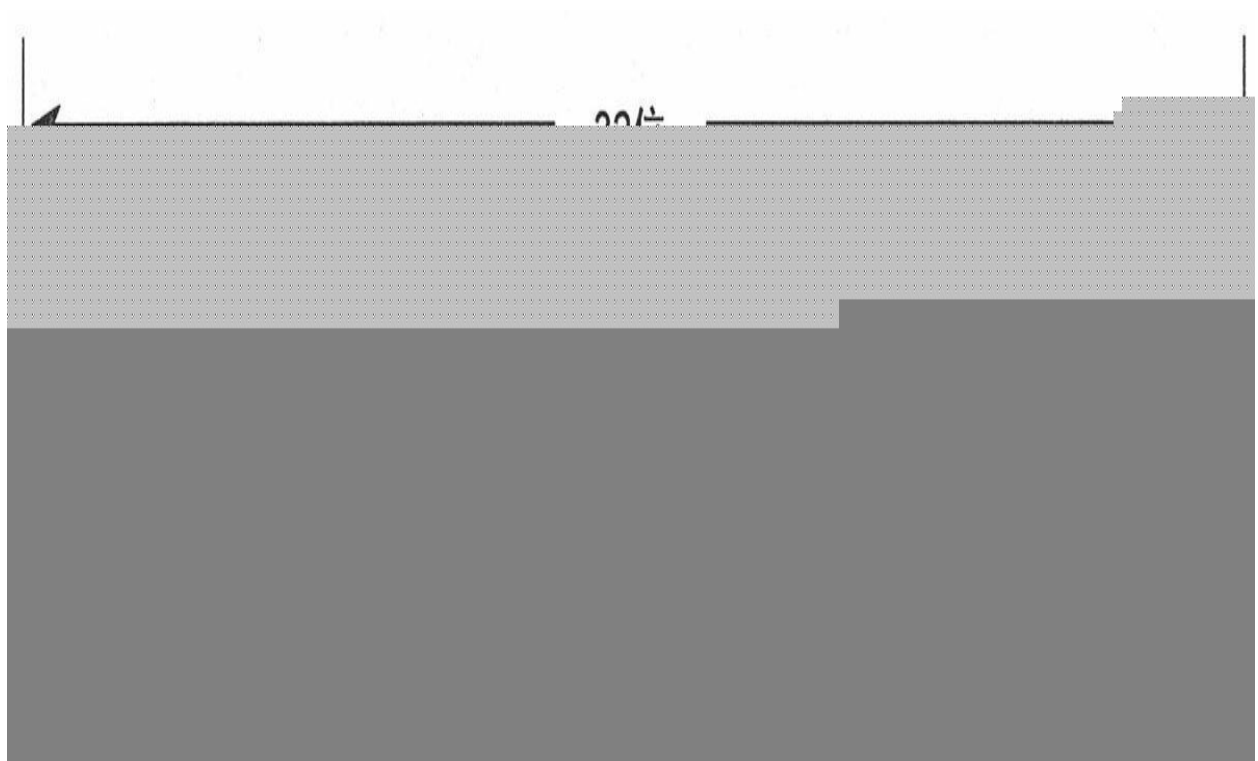
另外3台路由器仅把路由器Rembrandt作为它们各自的邻居来配置，并把优先级设置为10，这意味着路由器Rembrandt将成为DR路由器。通过使路由器Rembrandt成为DR路由器，这些PVC电路可以精确地仿效假设这4台路由器连接到一个广播型多路访问网络上时所形成的邻接关系。现在，OSPF数据包将以单播的方式转发到所配置的邻居地址上。

再次重申，**neighbor** 命令只在老的IOS版本（10.0版本以前）上才是必要的。而新的解决方案中，使用**ip ospf network** 命令可以改变缺省的OSPF网络类型。这条命令的一个选项是改变网络类型为广播型，这可以在每一个帧中继接口上使用**ip ospf network broadcast** 来实现。这个网络类型的更改将会让OSPF把这个NBMA视为一个广播型网络；在这种方案里，4台路由器的配置见示例8-79～示例8-82。

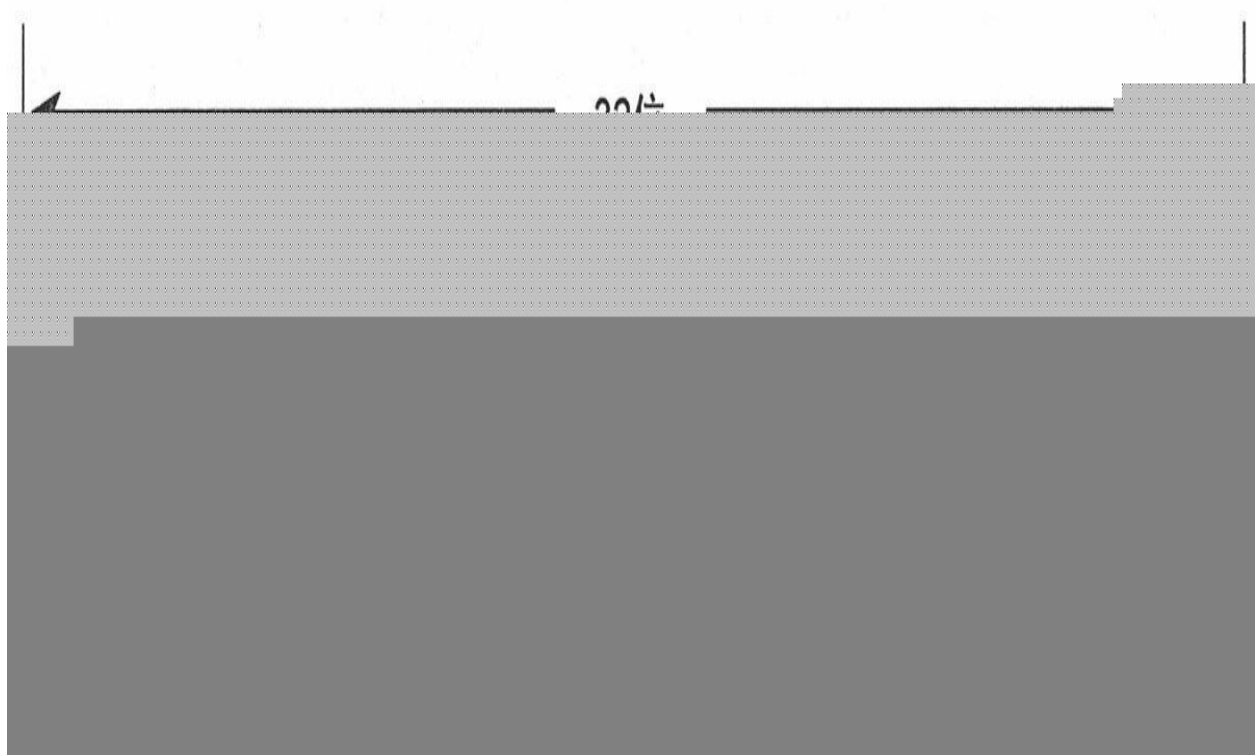
示例8-79 路由器Rembrandt的帧中继接口配置为一个OSPF广播型网络



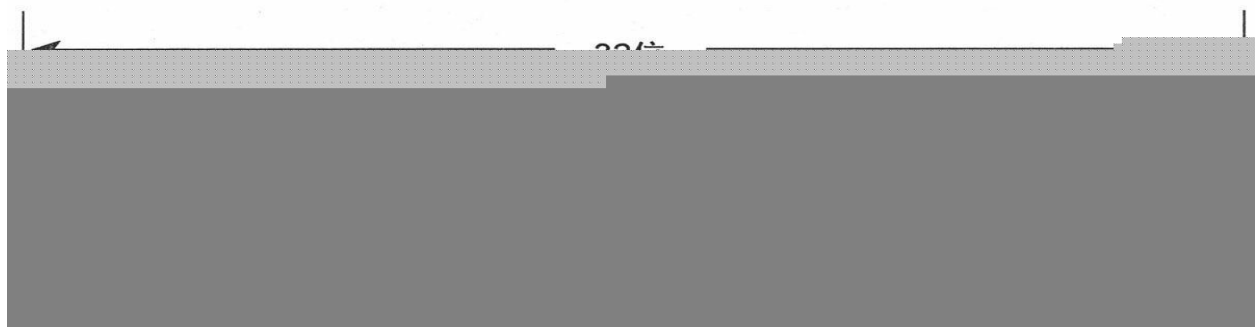
示例8-80 路由器**Hals**的帧中继接口配置为一个**OSPF**广播型网络

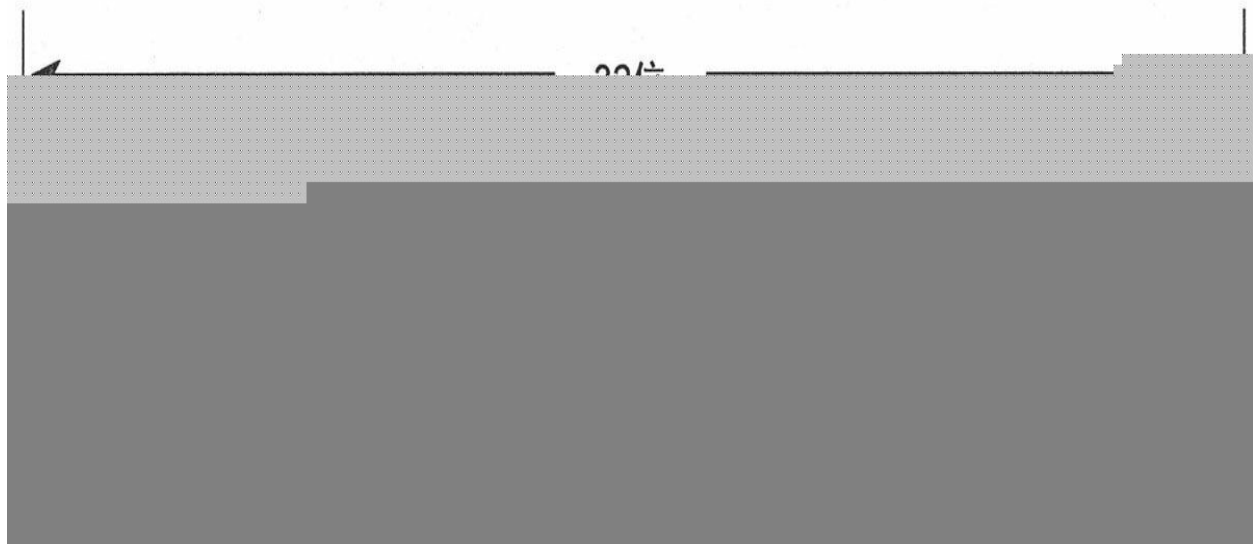


示例8-81 路由器**Vandyck**的帧中继接口配置为一个**OSPF**广播型网络



示例8-82 路由器Brueghel的帧中继接口配置为一个OSPF广播型网络





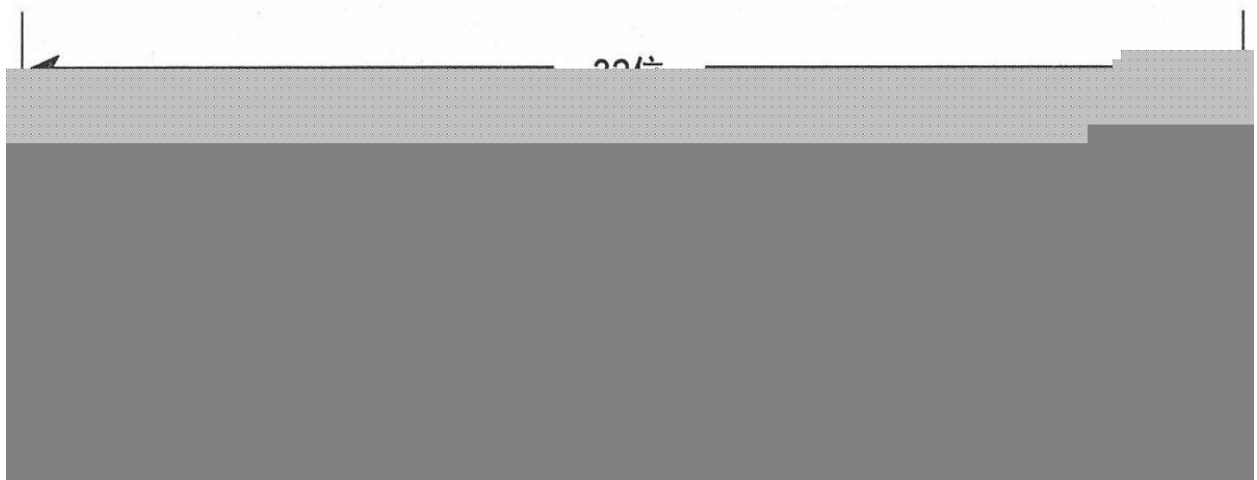
这里注意，在这个例子中，路由器Rembrandt的接口的优先级设置为10，而其他接口的优先级设置为0。这将可以再次确保路由器Rembrandt能够成为一台DR路由器。注意，静态的帧中继映射命令也设置为转发到广播和组播地址上了。

影响DR选取的另一种方案是实现一个全网状连接的拓扑结构，即每一台路由器都有一个PVC电路和其他所有的路由器相连。从路由器的角度来看，这种方案实际上是所有NBMA网络实现中最有效的方案。但是这种方案的一个显而易见的缺点就是开销相当昂贵。假设有 n 台路由器，那么为了创建一个全网状连接的拓扑结构将必须要有 $n(n-1)$ 条PVC电路才能实现。例如，图8-52中的4台路由器要创建一个全网状连接的拓扑应该需要6条PVC电路，而16台路由器则需要120条PVC电路。

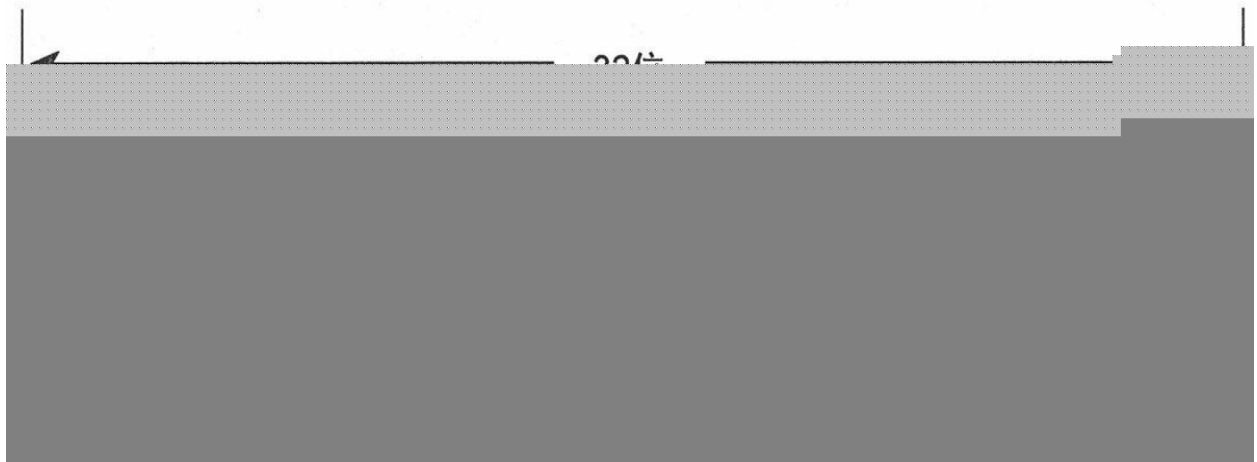
另外一种方法，就是将网络类型改为点到多点网络，这样就可以避免DR/BDR选取的处理。点到多点网络把PVC当作一个点到点链路的集合，因此就没有DR/BDR的选取发生。在多厂商的网络环境中，点到多点类型可能是广播型网络之外惟一的一种选择。

在示例8-83～示例8-86的配置中，把和每一个接口相关的OSPF网络类型改变成了点到多点类型。

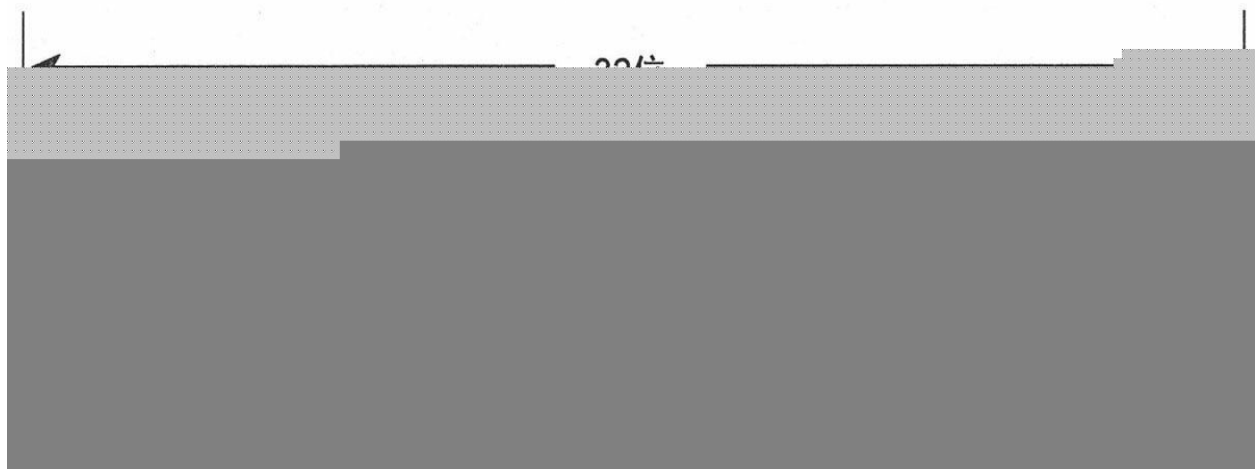
示例8-83 路由器Rembrandt的帧中继接口配置为一个OSPF点到多点型网络



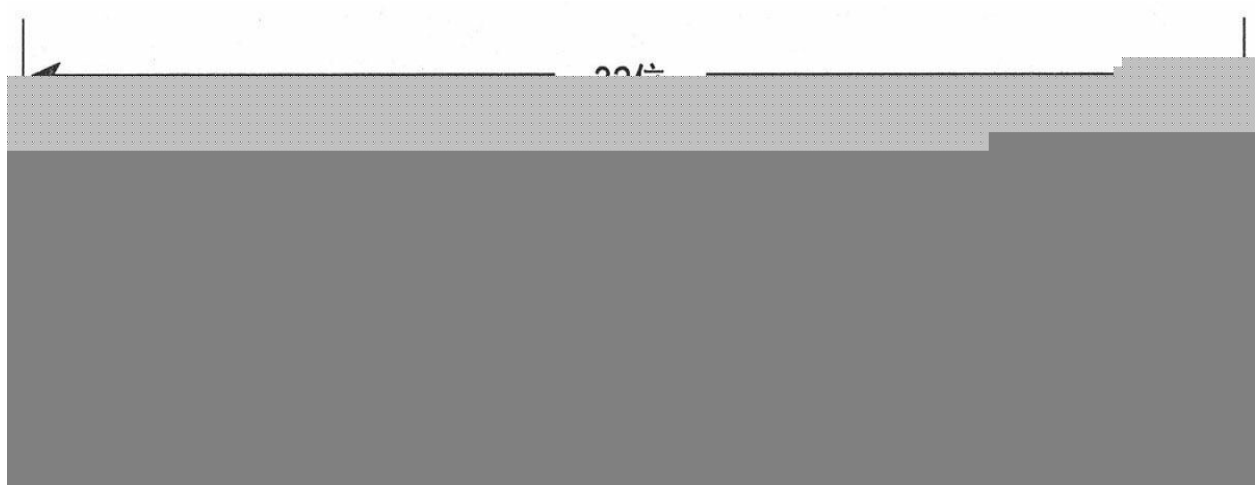
示例8-84 路由器Hals的帧中继接口配置为一个OSPF点到多点型网络



示例8-85 路由器Vandyck的帧中继接口配置为一个OSPF点到多点型网络



示例8-86 路由器Brueghel的帧中继接口配置为一个OSPF点到多点型网络

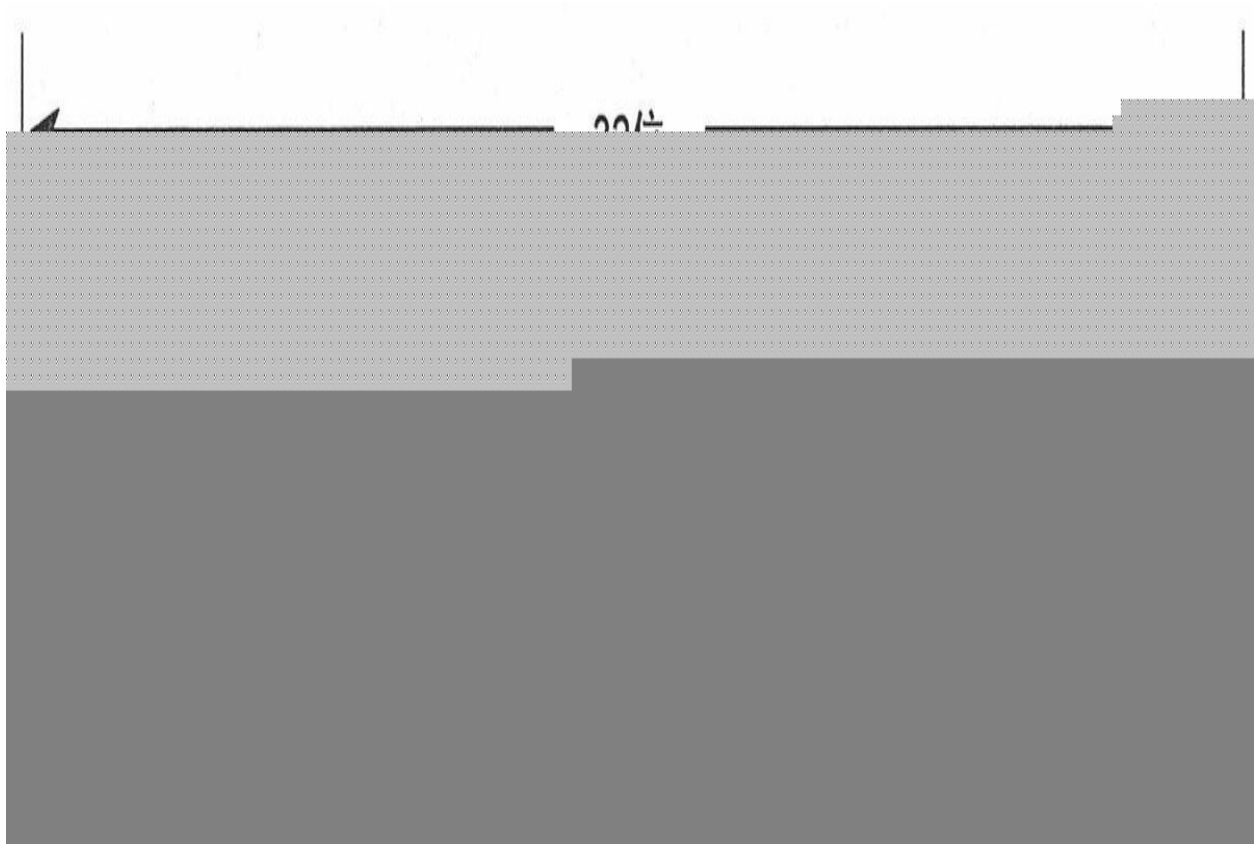


在这些配置中，利用帧中继网络逆向ARP解析功能动态地把网络层的地址映射到DLCI上，这种方法替代了上面例子中使用的静态映射命令。当然，如果想用，静态映射依然可以使用。

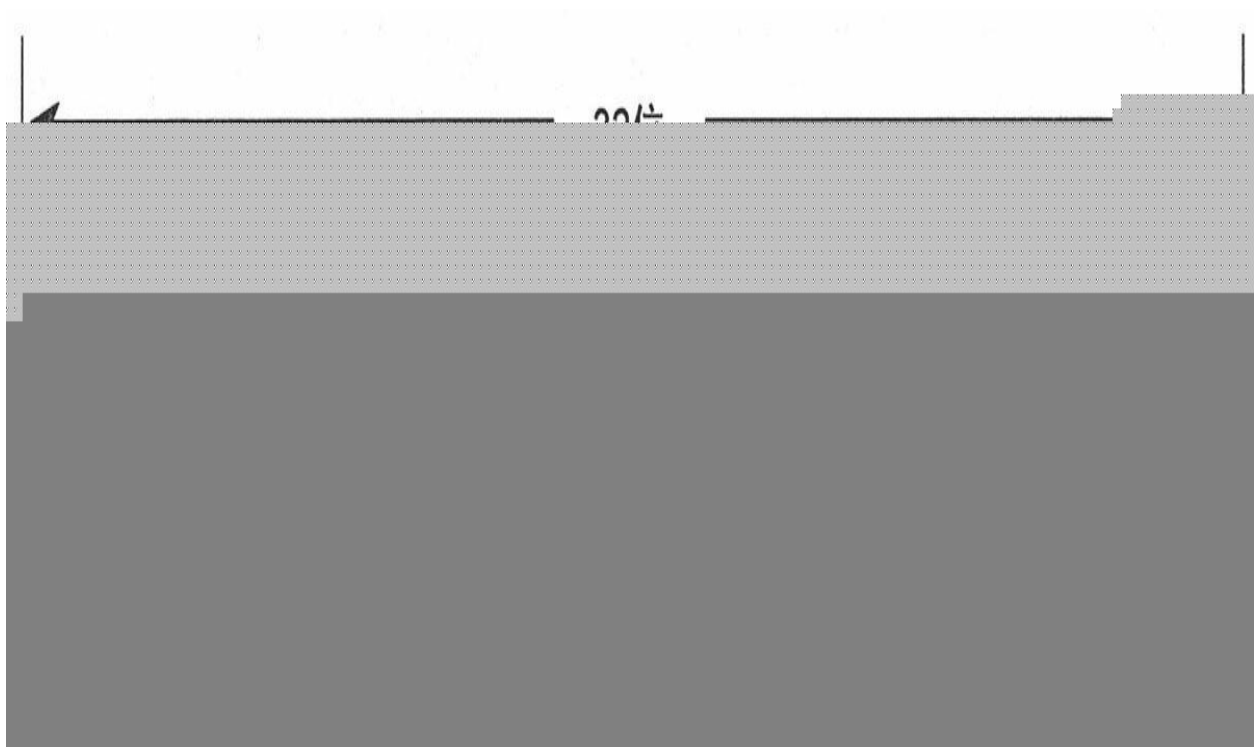
OSPF点到多点的网络类型把下层的网络看作一组点到点链路的集合，而不是一个多路访问网络，并且OSPF数据包以组播的方式发送到它的邻居。这种解决方案在那些动态连接的网络上会产生问题，像帧中继SVC或者ATM SVC等。自IOS 11.3AA开始，这个问题可以通过同时将一个网络声明为点到多点(point-to-multipoint)和非广播(non-broadcast)的方式而得到解决，配置参见示例8-87～示例8-90。

示例8-87 路由器Rembrandt的帧中继接口配置为一个OSPF点到多点

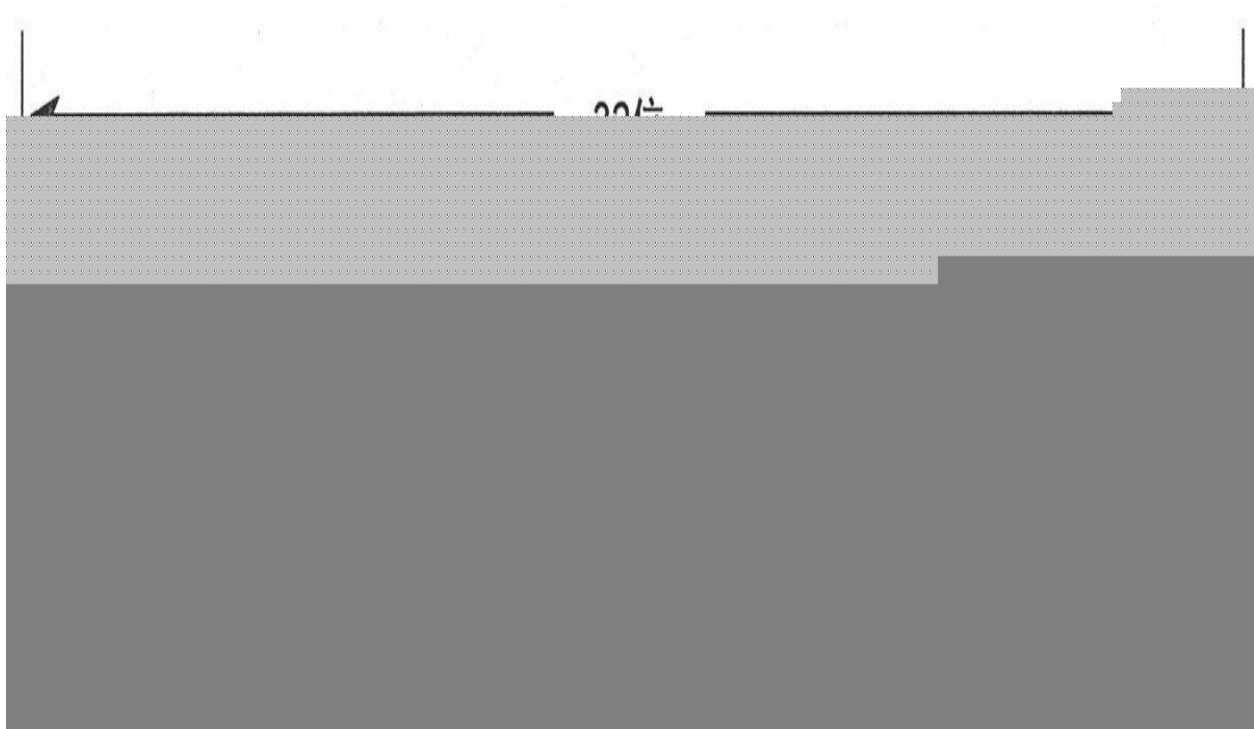
非广播网络



示例8-88 路由器**Hals**的帧中继接口配置为一个**OSPF**点到多点非广播网络

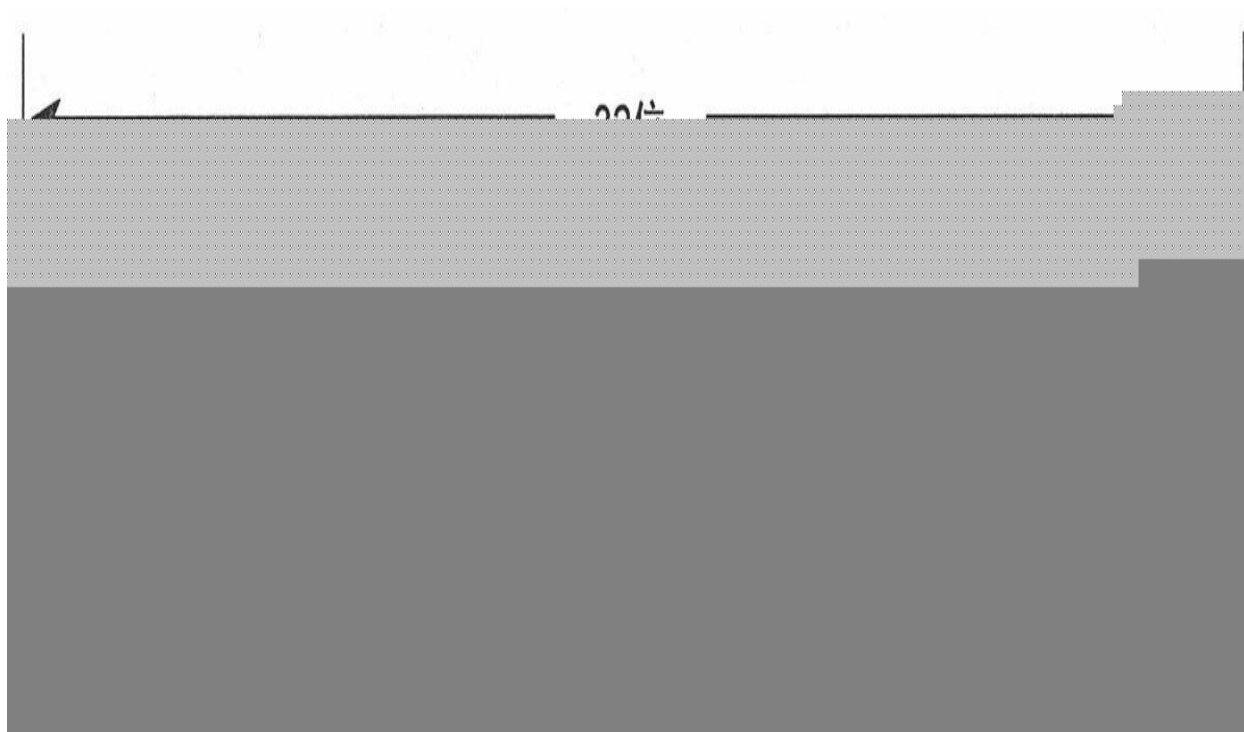


示例8-89 路由器**Vandyck**的帧中继接口配置为一个**OSPF**点到多点非广播网络



示例8-90 路由器**Brueghel**的帧中继接口配置为一个**OSPF**点到多点非

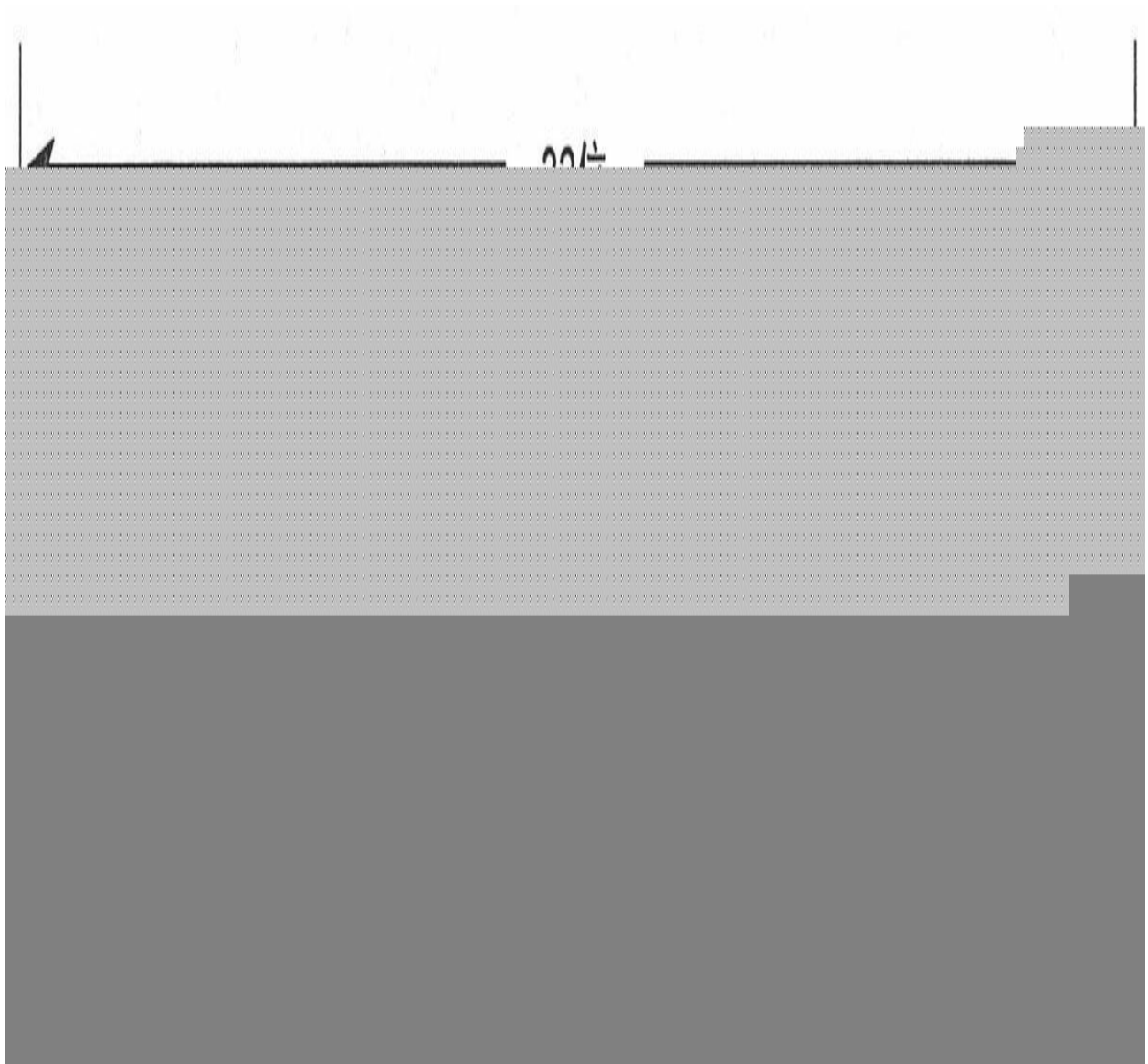
广播网络



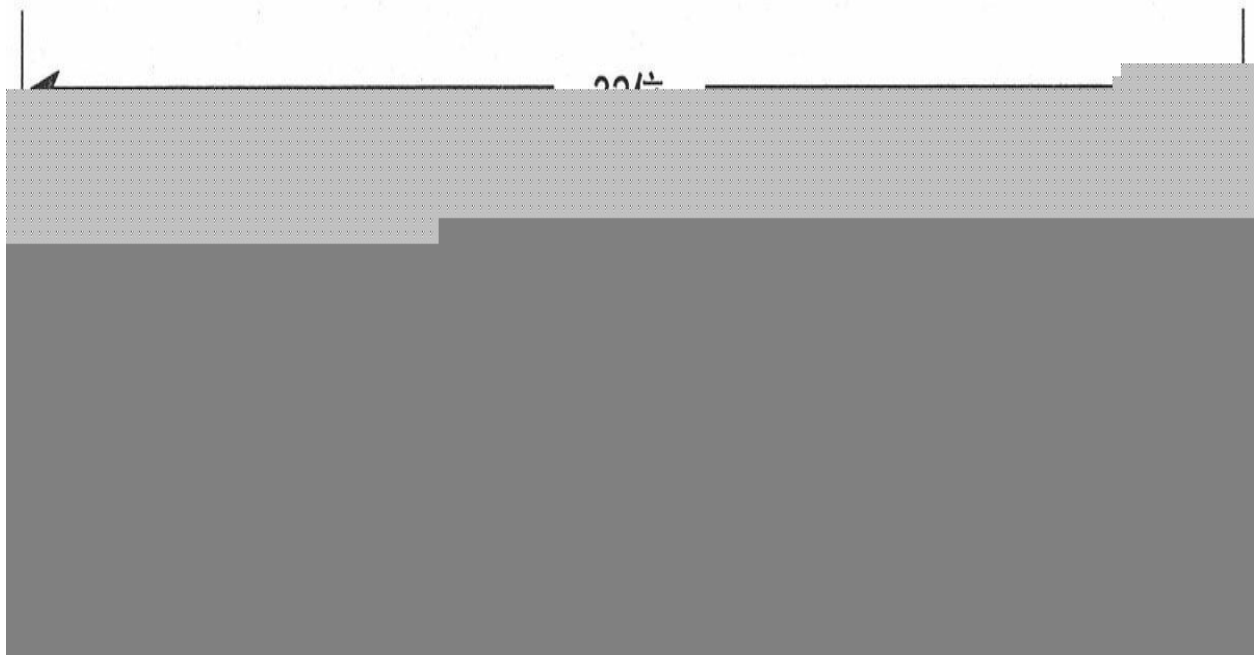
由于网络是非广播的，邻居将不会被自动地发现，因此必须要手工地配置。另一个在IOS 11.3AA中引入的特性可以从路由器Rembrandt的配置中看出来：即可以利用**neighbor** 命令基于每一个VC来指定它们的代价。

最后一种解决方案就是，使用它们各自本身的子网把每一个PVC连接作为一个单独的点到点网络，如图8-53所示。这种解决方案可以通过子接口来完成，配置见示例8-91～示例8-94。

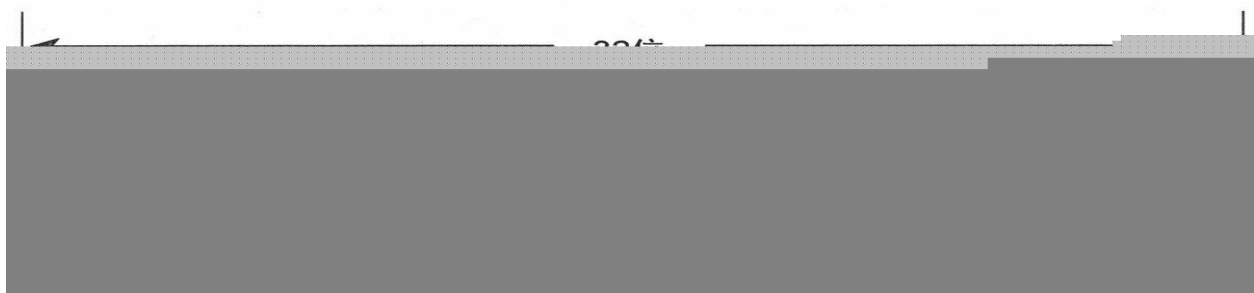
示例8-91 路由器Rembrandt配置了点到点子接口



示例8-92 路由器Hals配置了点到点接口



示例8-93 路由器Vandyck配置了点到点接口



示例8-94 路由器Brueghel配置了点到点接口

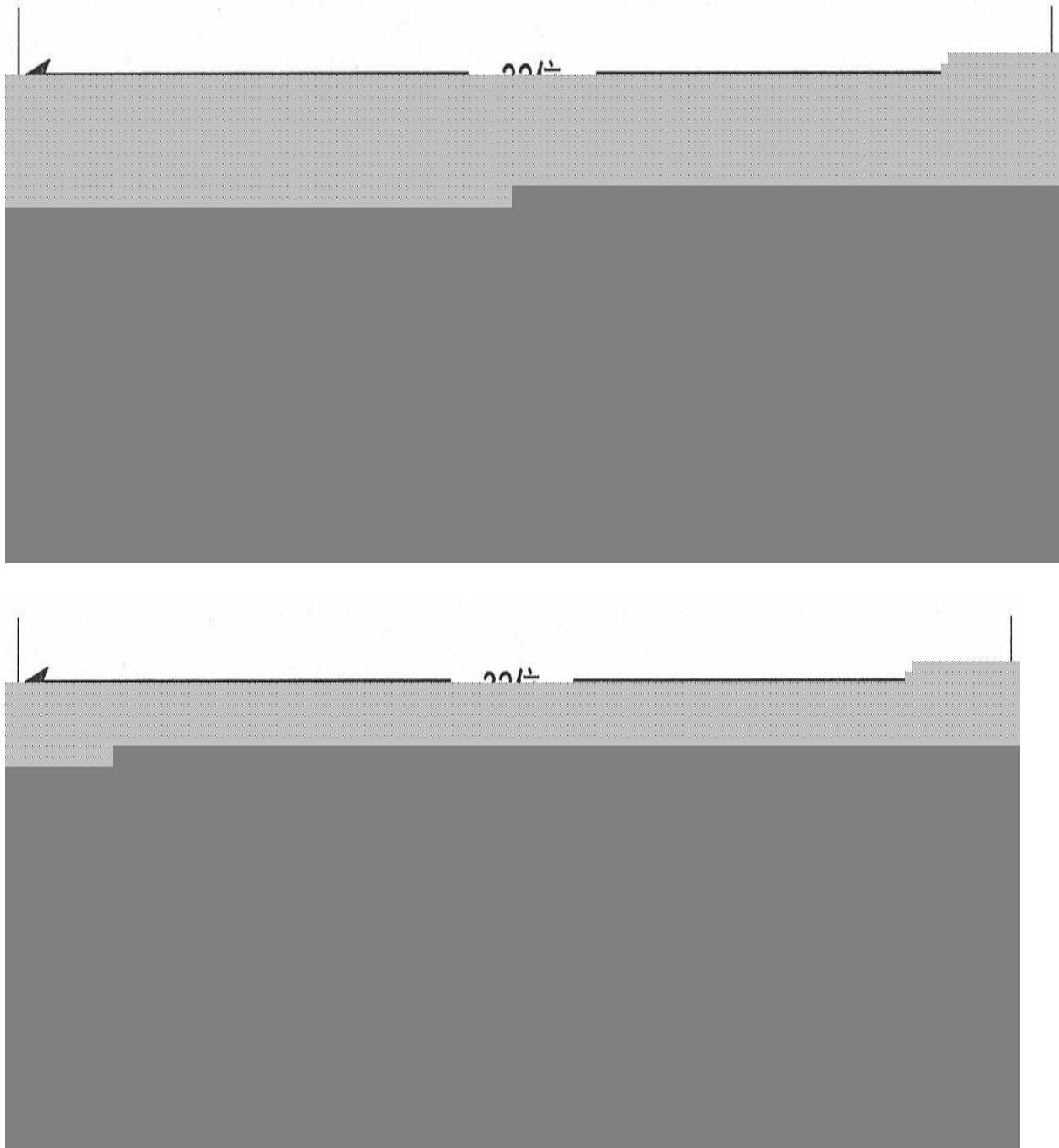


图8-53 点到点网络的子接口允许每一条PVC配置成一个单独的子网，并在NBMA网络上排除了DR/BDR的选取问题

对于所有运行在NBMA网络上OSPF配置来说，这种配置是最容易管理的。这种配置代码的一些优点是显而易见的，例如可以使用一个接口号对应于帧中继的DLCI，并且包括一个解释行。然而，还有一个主要的

优点就是可以在路由器之间建立简单的一对一的对应关系。

使用子接口有一个不太经常碰到的缺点是，每一个PVC电路都必须拥有自己的子网地址。在大多数情况下，这个需求不应该会产生问题，因此OSPF协议是支持VLSM的。正如这里的例子所显示的，可以从一个子网地址中创建更小的子网来分配给这个网络“云”是一件容易的事情。而且，由于PVC电路现在是点到点的链路，也可以使用无IP地址编号的方法作为子网地址需求的另一种选择。另一个重要的好处是路由器故障的检测。在点到点子接口上，点到点网络两端的路由器能够检测失效的虚电路（如果帧中继云提供足够的信号），如果存在另一条路由，路由选择协议也能够收敛到该路由上。

更需谨慎的是子接口会占用更多的内存。在一些内存有限的小型路由器上可能会给路由器带来较大的负担。

8.2.13 案例研究：运行在按需电路上的OSPF

配置运行在按需电路上的OSPF是很简单的，这可以在与按需电路相连的接口上，通过增加**ip ospf demand-circuit**命令来实现。而且只需要在点到点电路的一端，或者点到多点电路的多点端宣告为按需电路就可以了。在一般情况下，运行在按需电路上的OSPF不应该在一个广播型介质上实现。这是因为，这样的网络不能抑制Hello数据包的发送，从而使链路一直保持是活动(up)的。

如果图8-52中的虚电路是帧中继SVC电路，路由器Rembrandt的配置见示例8-95。

示例8-95 路由器Rembrandt配置为按需电路的OSPF

在按需电路上实现OSPF，需要记住以下几点：

- 只有在一个区域的链路状态数据库中的所有LSA都设置了DC-bit位以后，设置DoNotAge位的LSA才被允许进入该区域。这种设置方法可以确保该区域的所有路由器都具有识别DoNotAge位的能力。
- 实现按需电路上的OSPF的区域内的所有路由器都必须具有支持这种配置方式的能力。
- 如果运行在按需电路上的OSPF是在一个非末梢区域里实现的，那么在所有的非末梢区域内的路由器都必须支持这种方式。原因是DC-bit位的设置是在类型5的LSA中实现的，而这种类型的LSA是泛洪扩散到所有的非末梢区域中的。
- 应该尽量限制在末梢区域、完全末梢区域或NSSA区域上实现按

需电路的OSPF。这种实现方式可以取消要求OSPF域内的所有路由器支持按需电路上的OSPF的需要。这种方式也将因其他区域拓扑改变而引起改变的LSA的数量减到最小，并可以防止过多地使按需电路处于活动状态。

- 如果配置了按需电路上的OSPF并且配置了一条虚链路要穿过这条按需电路，那么这条虚链路也将被看作是一条按需电路。另外，这条虚链路的通信流量将会保持按需电路为活动(up)。
- 每隔30min，OSPF协议就会重新刷新一次它的LSA，以防止这些LSA驻留在链路状态数据库时变得无效。由于设置DoNotAge位的LSA在穿过按需电路的时候不会重新刷新，因此也就丧失了OSPF的这个稳定特性。
- 重新刷新过程可以在一条按需电路两端外的其他所有接口上发生，但是LSA在通过这条按需电路时不能进行重新刷新。结果是，这条链路两端的同一个LSA的各自序列号可能是不同的。网络管理工作站可以使用某些MIB变量[\[33\]](#)去验证数据库的同步；如果序列号在数据库中不匹配，那么将会错误地报告一个错误。

8.3 OSPF故障诊断

OSPF协议的故障排除有时是令人恐怖的，这在一个大型网络上显得尤其明显。但是，OSPF产生的路由选择问题和其他任何路由选择协议的路由选择问题没有什么不同，故障可能是下列某种原因之一引起的：

- 路由信息丢失；
- 错误的或不精确的路由信息。

对路由器的检查仍然是获取故障排除信息的主要来源。使用命令`show ip ospf database`查看不同的LSA也会得到重要的信息。例如，如果一条链路是不稳定的，那么通告它的LSA将会频繁变化。这种情况反映在它的序列号会比其他LSA的序列号明显的偏高。网络不稳定的另外一个迹象是LSA的老化时间从来不会变得很大。

请记住，一个区域内所有路由器的链路状态数据库都是相同的。因此，除非你怀疑某些路由器的链路状态数据库本身变得有问题，否则就可以通过检查某一台路由器的链路状态数据库来检查整个区域的链路状态数据库。另外一个好的经验是，为每一个区域的链路状态数据库做一个拷贝（软拷贝或硬拷贝）。

当检查单独一台路由器的配置时，需要考虑以下几点：

- 所有接口配置的地址和掩码是否正确？
- **network area** 语句使用的反向掩码是否正确？是否匹配正确的接口？
- **network area** 语句是否把所有的接口都指定到正确的区域中了？
- **network area** 语句的使用顺序正确吗？

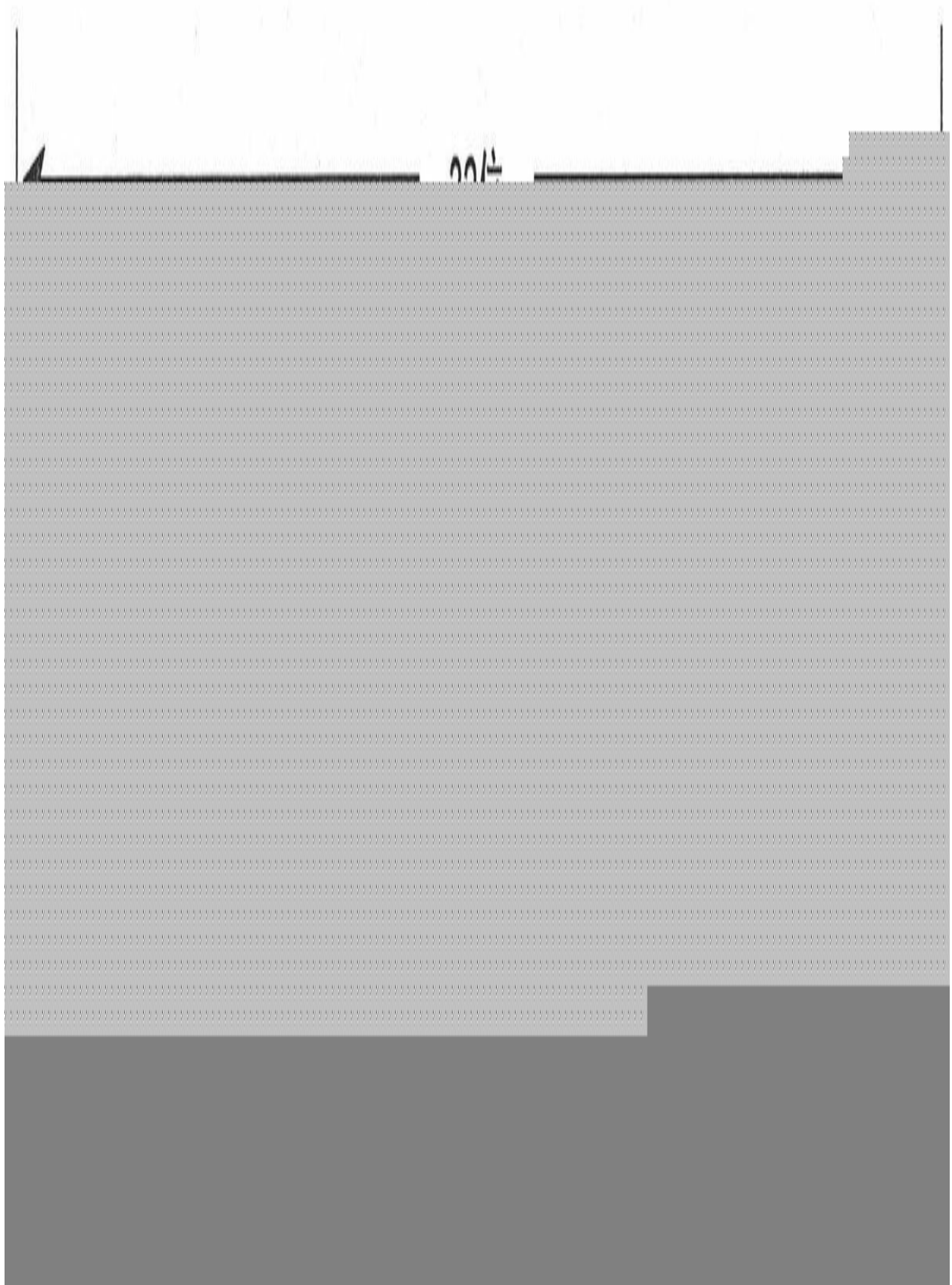
当检查邻接关系时（或者缺少邻接关系时），请考虑下面这些问题：

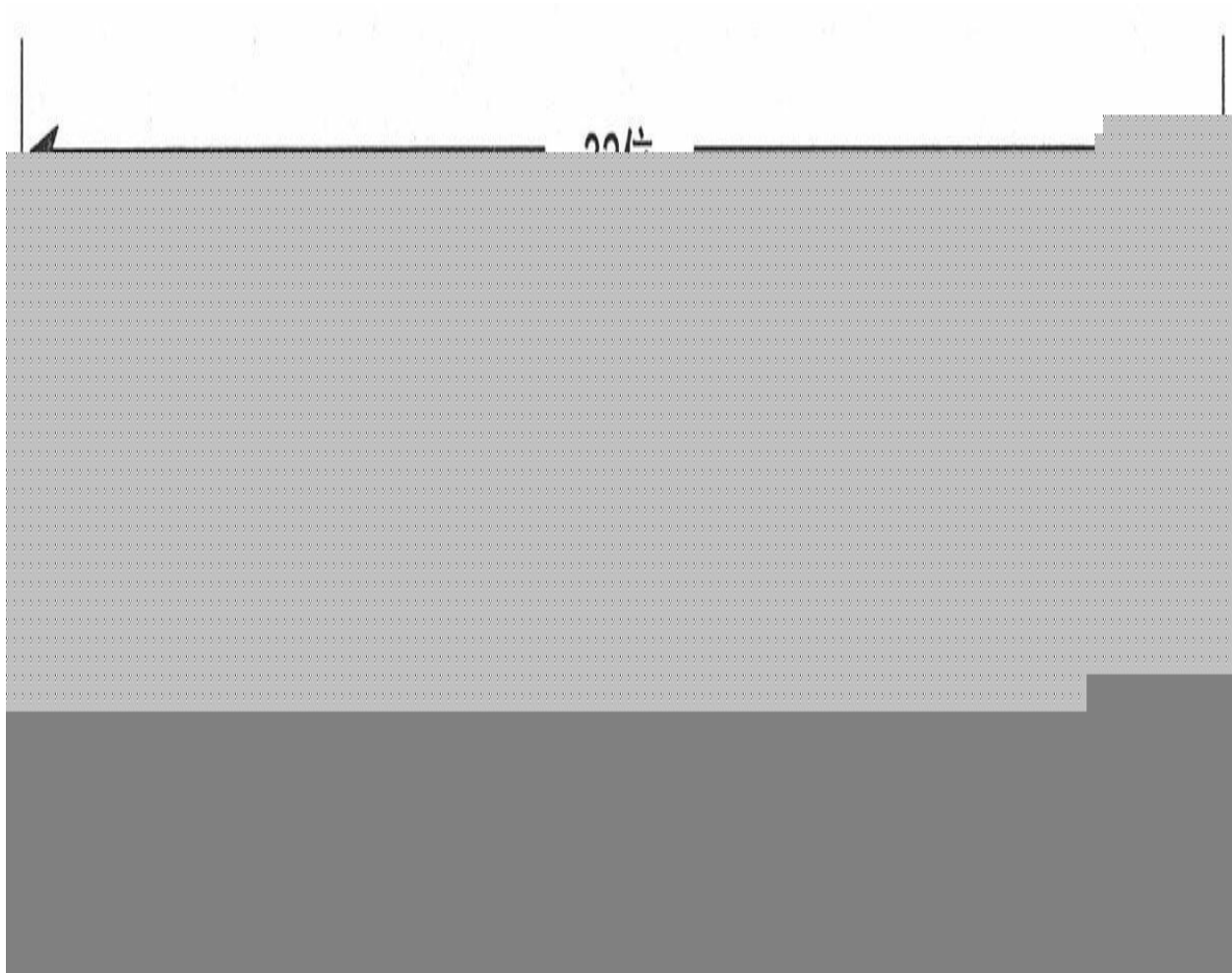
- 从这两台邻居路由器中有Hello数据包正在发送吗？

- 这两个邻居路由器之间的计时器设置相同吗？
- 这两个邻居路由器之间的可选性能字段设置相同吗？
- 接口是配置在同一个子网上的吗（也就是说，它们的地址 / 掩码对是否属于同一个子网）？
- 邻居路由器的接口是否是同一种网络类型？
- 一台路由器是否正在试图和它的邻居路由器的辅助地址形成一种邻接关系？
- 如果使用了认证，那么在邻居路由器之间的认证类型是否相同？口令和密钥（在MD5的实例中）是否相同？该区域内的所有路由器是否都启用了认证？
- 在所有的访问列表中，是否有正在阻塞OSPF的访问列表？
- 如果邻居关系穿过一条虚链路，那么这条链路是否配置在一个末梢区域内？

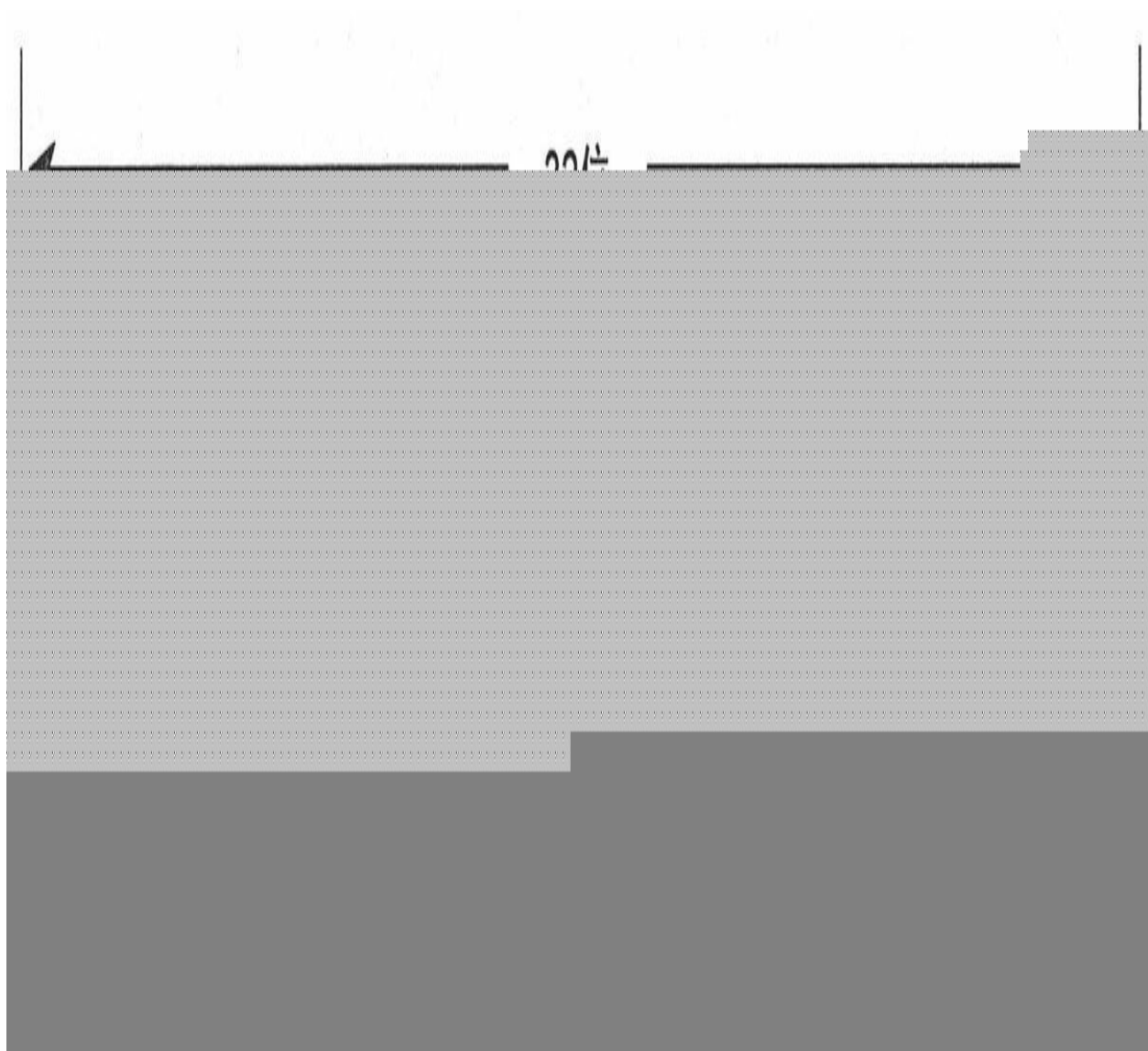
如果怀疑某个邻居或者某个邻接关系变得不稳定了，那么可以通过 **debug ip ospf adj** 命令来监控这些邻接关系。然而，这条命令经常会产生比所需要的信息多得多的内容，参见示例8-96。邻居的状态变化被非常详细的记录下来。如果记录了普通的Hello数据包处理。如果在一个扩展期间执行了监控，这些正常的信息将会溢出路由器的内部缓冲区。从IOS11.2版本开始，路由器可以在它的OSPF配置里增加命令 **log-adjacency-changes [detail]** 来监控邻接关系。这条命令将会记录邻接关系变化的一个简单日志，参见示例8-97和示例8-98。

示例8-96 使用命令 **debug ip ospf adj** 输出的调试信息显示，当一台邻居路由器的以太网接口暂时断开随后又重新连接上的结果

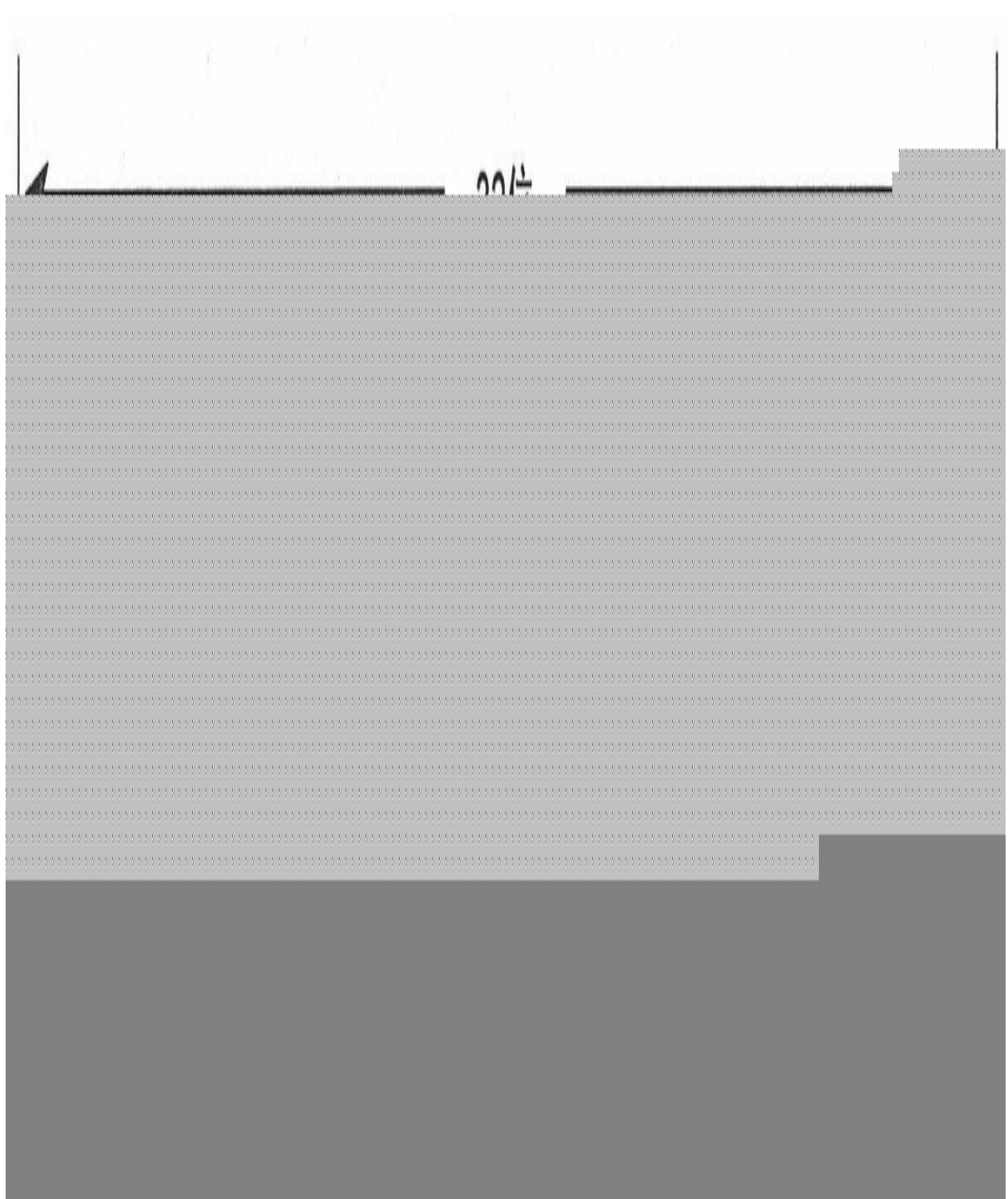




示例8-97 这些日志信息的记录结果来自命令**log-adjacency-changes**，显示了示例8-96中相同的邻居失效的信息，但是没有上面的调试信息那么详细



示例8-98 这些日志信息的记录结果来自OSPF配置命令**log-adjacency-changes**，显示了示例8-96中相同的邻居失效的信息，但比示例8-97中的记录更加详细



如果怀疑一个链路状态数据库出现问题，或者这两个数据库不同步，那么可以使用命令**show ip ospf database database-summary** 来观察每一台路由器中数据库的LSA数量。对于给定的一个区域，在所有的路由器上，每一种LSA类型的数量都应该是相同的。下一条命令**show ip ospf database** 将显示一台路由器数据库的每一条LSA的校验和。在一个给定的区域内，在每一台路由器的数据库中的每一条LSA的校验和都应该相同。除非是在一个非常小的数据库里，否则校验这个情况的正确与否将非常乏味和痛苦。不过幸运的是，还有MIB [\[34\]](#)，MIB可以在一个SNMP网络管理平台上报告一个数据库校验和的总和。如果在一个区域的所有数据库都同步，那么每一个数据库中的总和都应该是相同的。

当检查一个区域层面上的问题时，请记住以下几个问题：

- ABR路由器是否配置正确？
- 对于相同区域的类型是否所有的路由器都配置了？例如，如果一个区域是末梢区域，那么所有的路由器都必须使用**area stub**命令。
- 如果配置了地址汇总，那么配置的正确吗？

如果是路由器的性能出现了问题，那么可以在路由器上检查它们的CPU和内存的使用情况。如果内存的使用率在70%以上，那么可能是链路状态数据库太大了；如果CPU的利用率一直保持在60%以上，那么网络的拓扑可能存在不稳定的情况。如果内存或CPU的利用率超过了50%的警戒线，网络管理员就应该开始分析网络性能加重的原因，并基于得出的分析结果，来制定改善网络的升级计划。

末梢区域和地址汇总能够帮助减小链路状态数据库的大小并能容忍网络的一些不稳定性。最能加重一台OSPF路由器负担的是对LSA的处理，而不是SPF算法的计算。在个别情况下，类型1和类型2的LSA对处理器的影响比汇总LSA更大。但是，类型1和类型2的LSA可以编成组发送，而汇总LSA却只能在单个数据包中发送。结果是，汇总LSA实际上对处理器的影响更大。

下面的案例研究演示了进行OSPF协议的故障排除时，使用最频繁的技巧和工具。

8.3.1 案例研究：孤立的区域

区域内的数据包可以在图8-54中的区域1内进行路由转发，但是所有的区域间通信的尝试都失败了。这种情况下，首先应该马上怀疑是区域1的ABR路由器出现了故障。而且由于内部路由器没有关于ABR路由器的入口，这使得更加坚信可能是ABR路由器出现了问题（参见示例8-99）。

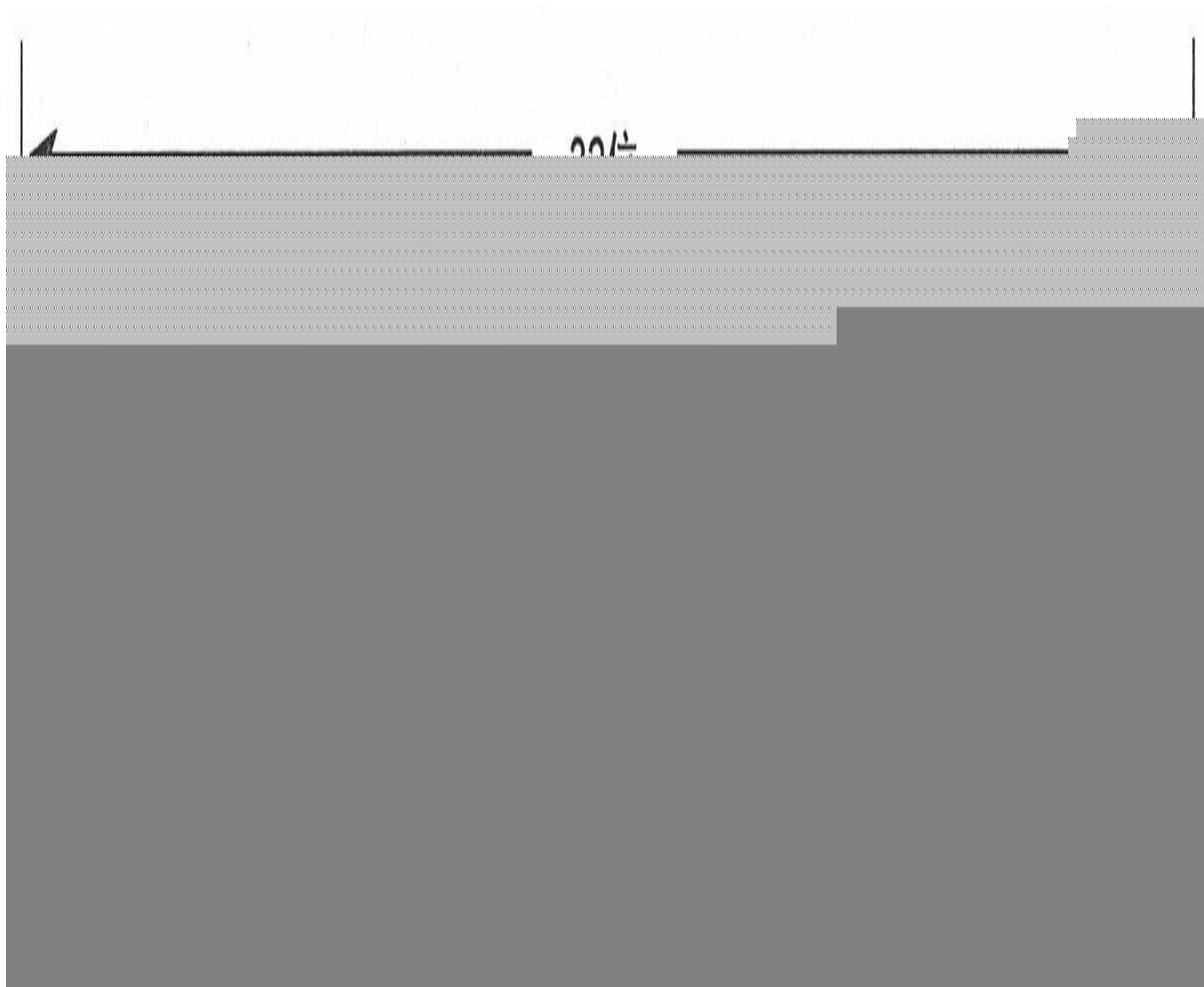
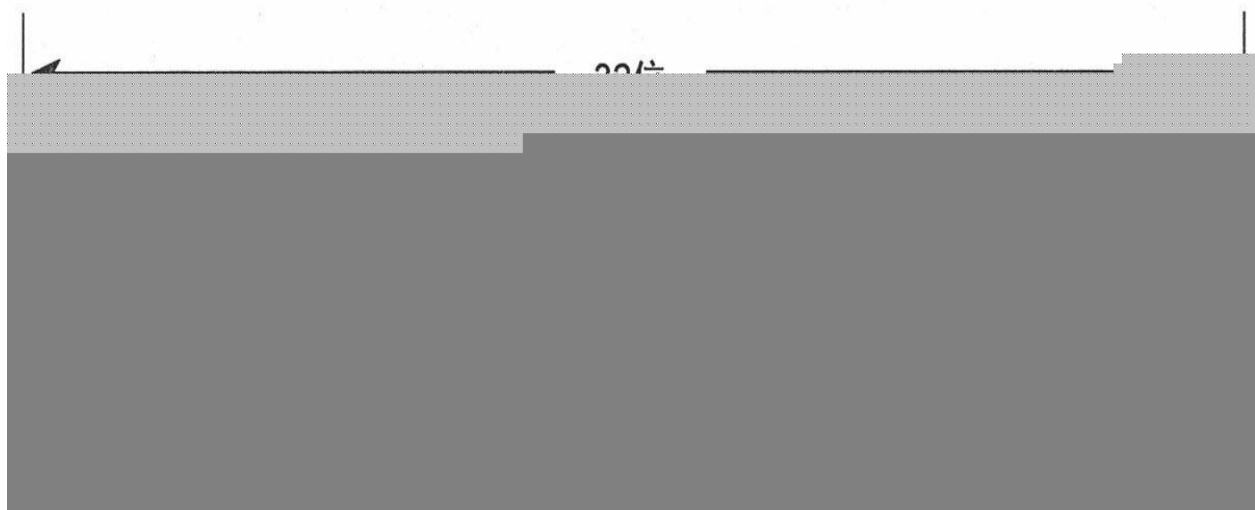


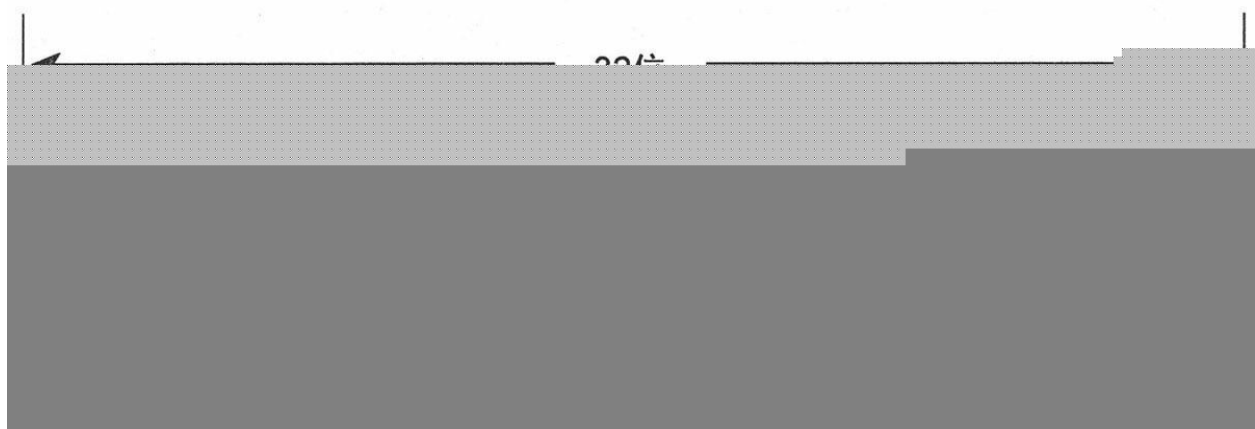
图8-54 区域1内的系统和路由器通信正常，但是没有通信流量可以通过或者来自区域0

示例8-99 使用命令**show ip ospf border-router**可以检查内部路由器的内部路由表信息。没有显示关于**ABR**路由器的路由器条目



下一步是验证连接到ABR路由器的物理链路是否正常以及OSPF协议是否在正常工作。参见示例8-100所示，在上述的那一台内部路由器的邻居表中，显示了关于ABR路由器的邻居状态都是完全邻接的，这表明邻接关系是存在的。事实上，这台ABR路由器是这里令牌环网络的DR路由器。邻接关系的存在证实链路是正常的，而且也可以交换含有正确参数的OSPF Hello数据包。

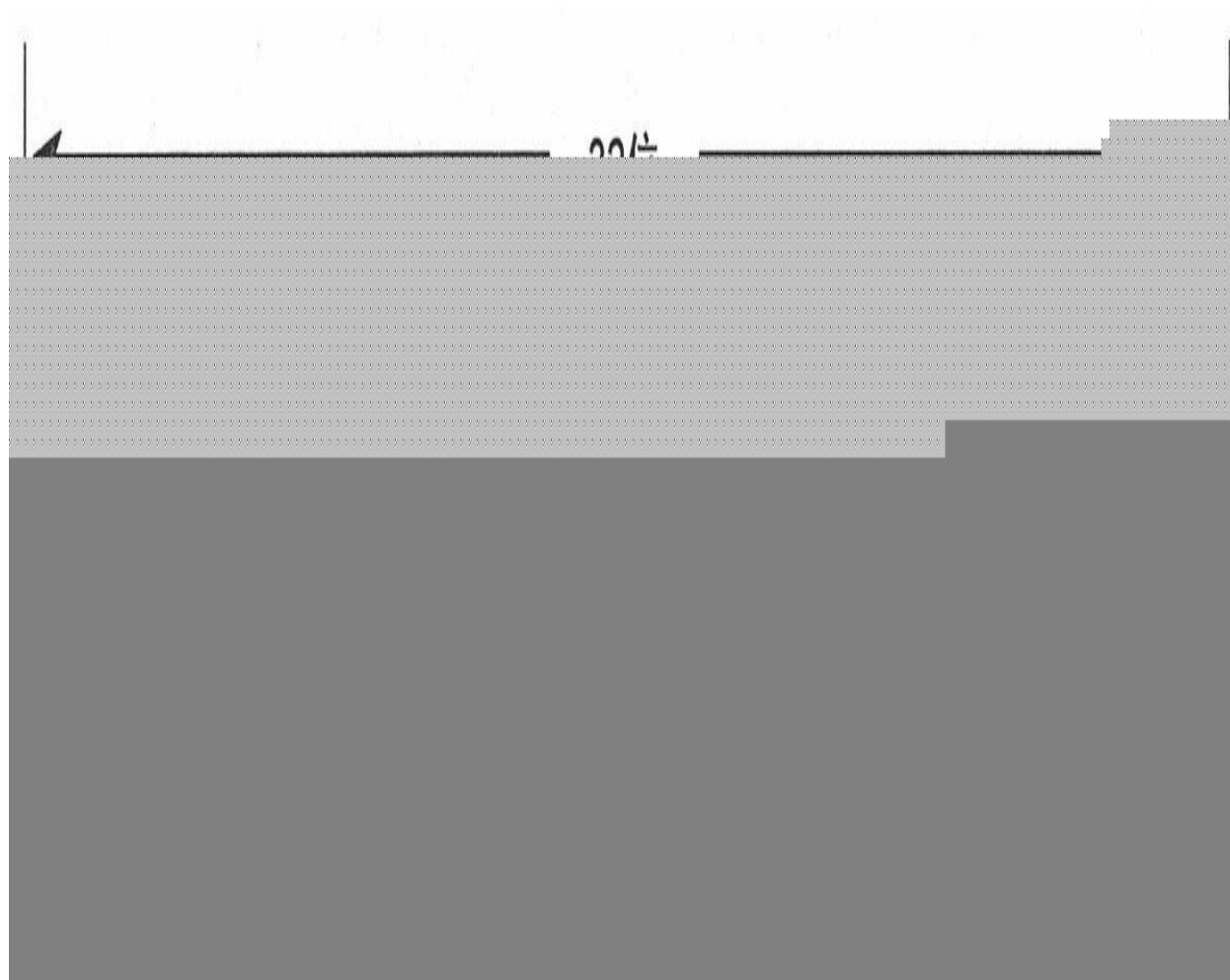
示例8-100 路由器National的邻居表表明与ABR路由器(1.1.1.1)的邻接关系是完全邻接的



在路由器National的链路状态数据库和它的路由表中可以发现一些其他的和这个故障有关的迹象。在示例8-101中，路由器National的数据库中只包含了路由器LSA（类型1）和网络LSA（类型2），但是没有记录通告区域外部目的地址的网络汇总LSA(类型3)。同时，出现了由路由器Whitney(1.1.1.1)始发的LSA。这个信息再次表明路由器Whitney与路由器

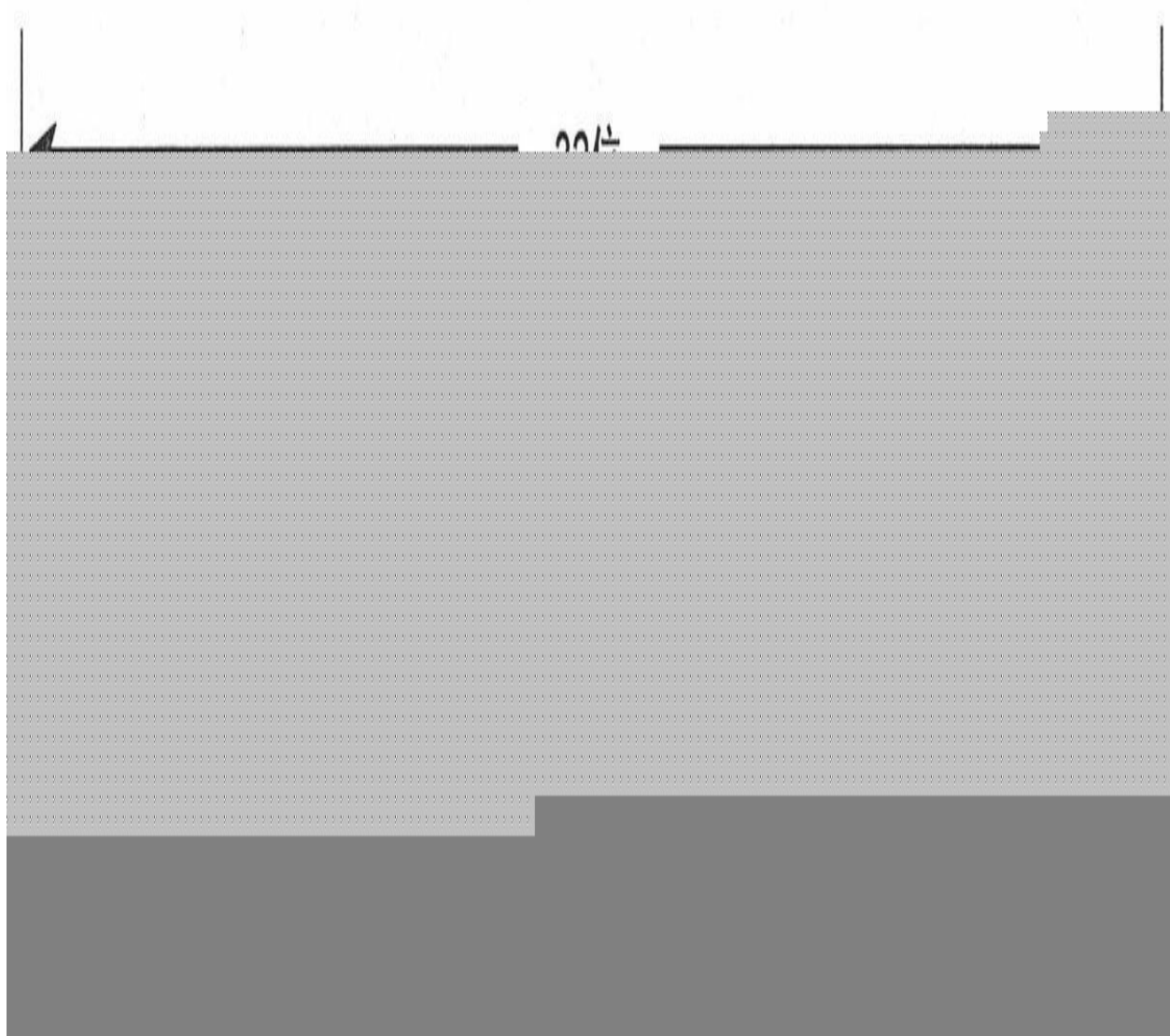
National是有邻接关系的，但是却没有信息从区域0通过到达区域1。

示例8-101 路由器**National**的链路状态数据库显示，和路由器**Whitney**之间是有邻接的，但是却没有通告区域间的目的地址



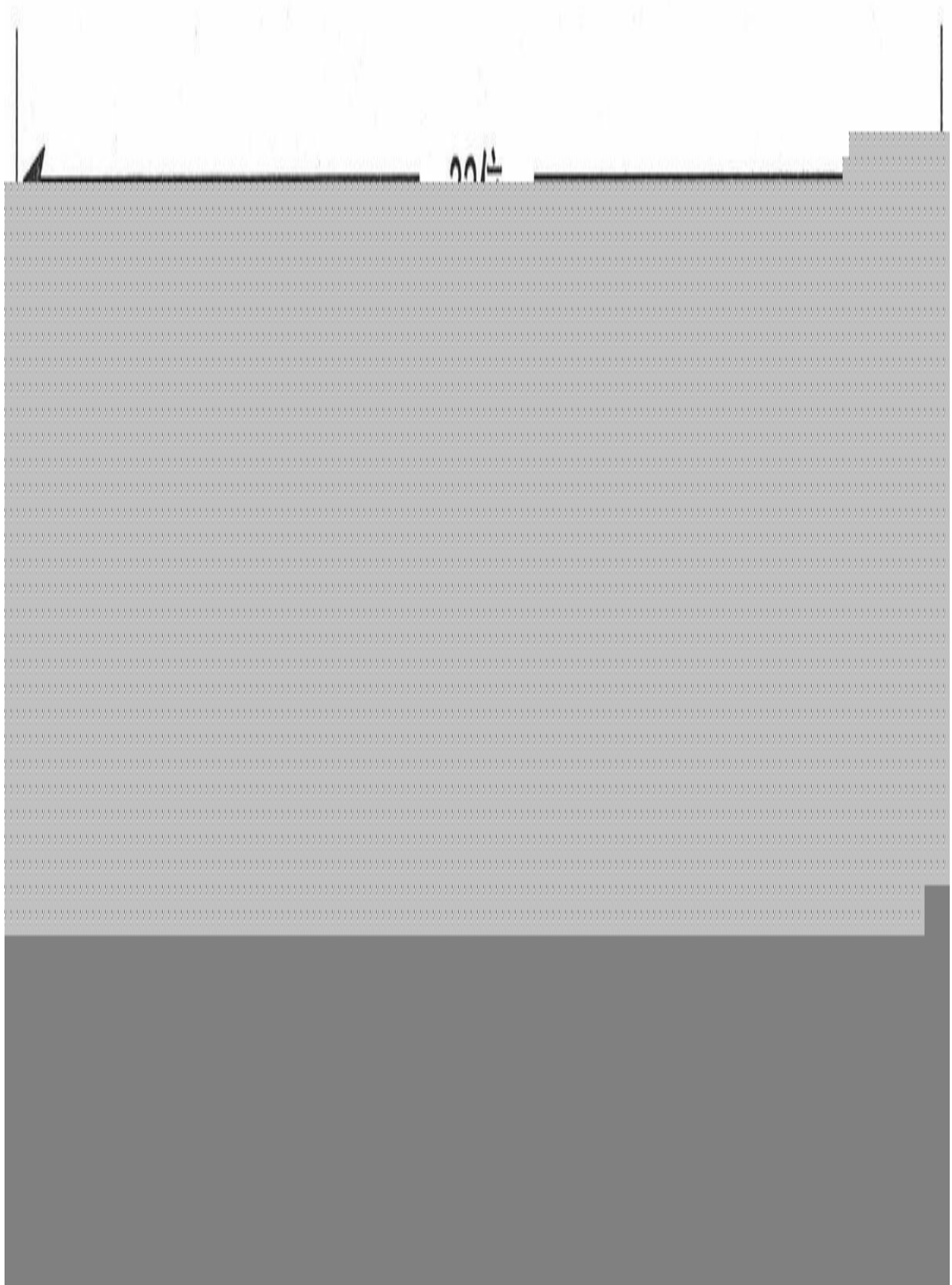
参见示例8-102，在路由器**National**的路由表中，区域1外部惟一的目的地地址是与路由器**Whitney**相连的串行链路地址。然而，在这里还显示了另外一条线索：这些路由条目被标记成区域内路由(O)。依照图9-54，这些路由应该是在区域0内的，因此，它们应该是被标记成区域间路由(OIA)。现在，问题显然出在ABR路由器的区域0的一端。

示例8-102 路由器**Whitney**正在通告它的串行接口的子网，但是它们正在被作为区域内的目的地址通告



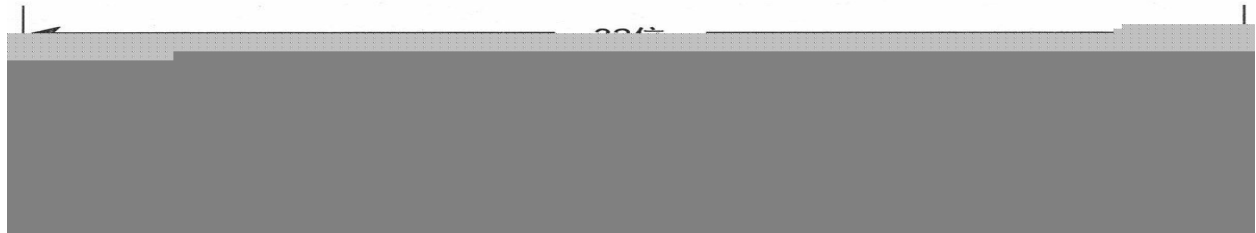
参见示例8-103，虽然还没有发现问题产生的最终原因，但是通过对路由器Whitney的串行链路的检查已经发现了问题所在。这两个串行接口应该在区域0内，但都被区域1替代了。它们都和网络逻辑拓扑上的邻居（路由器Louvre和Hermitage）相连，但是却没有记录OSPF邻居。有规律显示的错误信息表明路由器Whitney正在接收来自路由器Louvre和Hermitage的Hello数据包，而这些数据包的区域字段设置为0，因而引起不匹配的情况发生。

示例8-103 路由器Whitney的串行接口被配置成区域1，从而替代了区域0；这个配置在接收区域0的Hello数据包时会引起错误信息



路由器Whitney的OSPF配置参见示例8-104。

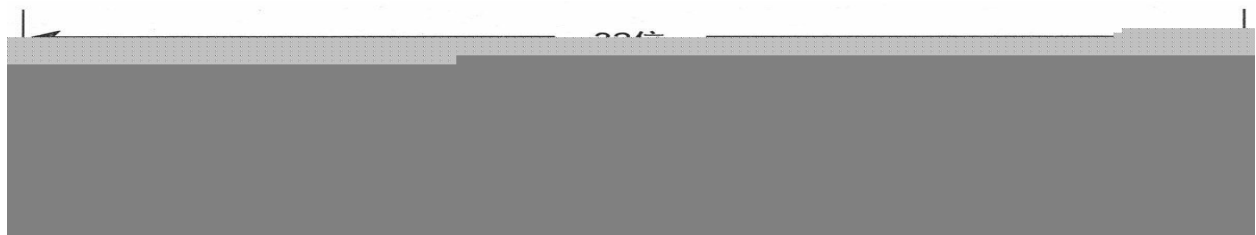
示例8-104 路由器Whitney的OSPF配置



乍一看，这个配置好像是没有问题的。但是，回忆一下第一个案例研究中提到的，**network area** 命令是连续执行的。第二条**network area** 命令只可能影响到那些和第一条**network area** 命令不匹配的接口。在这样的配置下，所有匹配第一条**network area** 命令语句的接口都被设置到区域1了。而第二条命令并没有被应用。

正确的配置显示在示例8-105中。

示例8-105 路由器Whitney正确的OSPF配置



当然，这里可以有多种有效的正确配置。但重要的一点是第一条**network area**命令必须足够精确地只去匹配区域1的地址，而不含有区域0接口的地址。

8.3.2 案例研究：路由汇总配置错误

如图8-55所示，显示了一个骨干区域和与之相连的3个区域。为了减小链路状态数据库的大小和增强网络的稳定性，在区域之间使用了路由汇总。

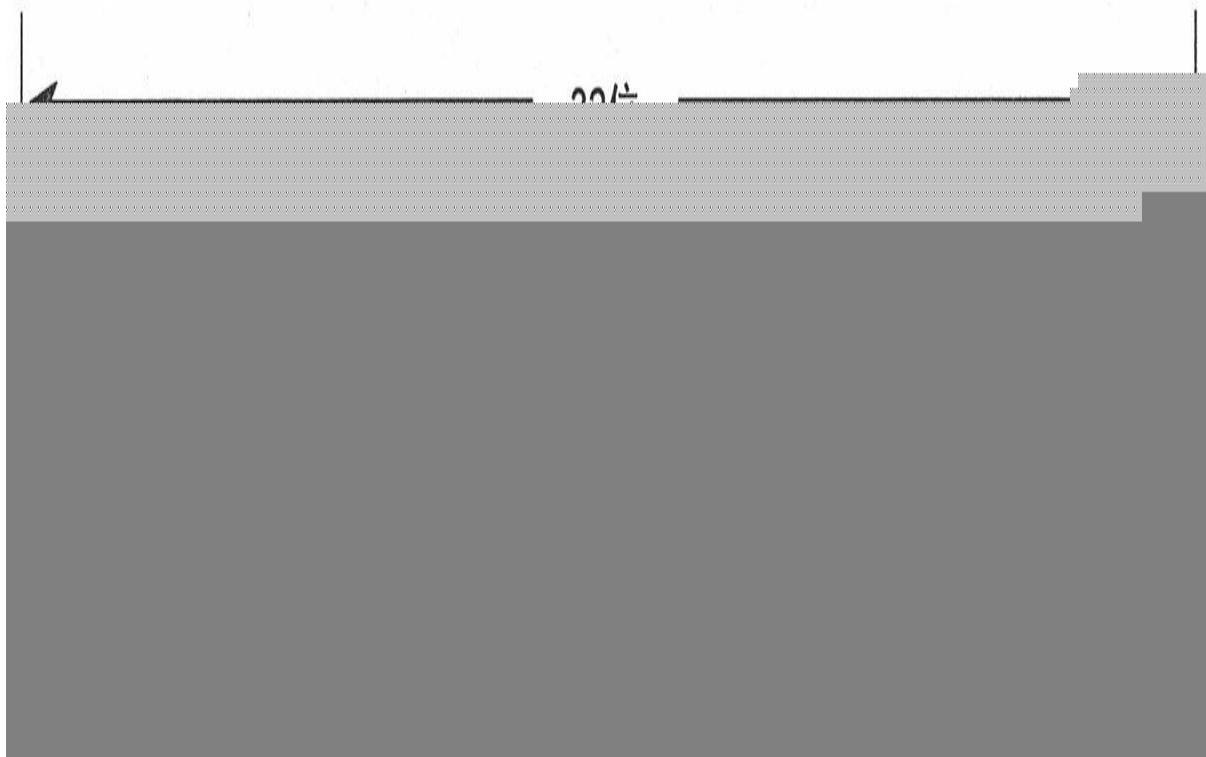


图8-55 这里显示了被通告到区域0内的每一个区域的汇总地址。区域0的子网地址也被汇总到其他的区域中

图8-55中显示的每个区域的地址汇总了这3个非骨干区域内的个别子网。例如，区域1内的一些子网可能是：

172.16.192.0/29

172.16.192.160/29

172.16.192.248/30

172.16.217.0/24

172.16.199.160/29

172.16.210.248/30

图8-56中显示了这些子网地址能够被汇总成地址172.16.192.0/19。

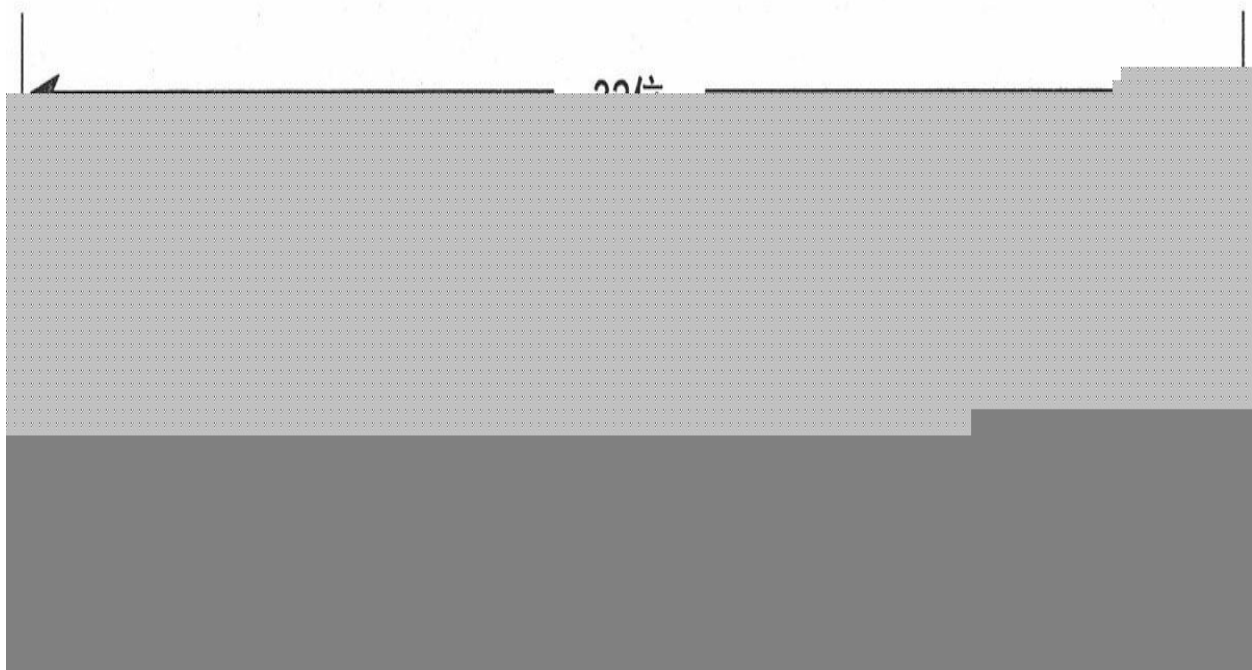
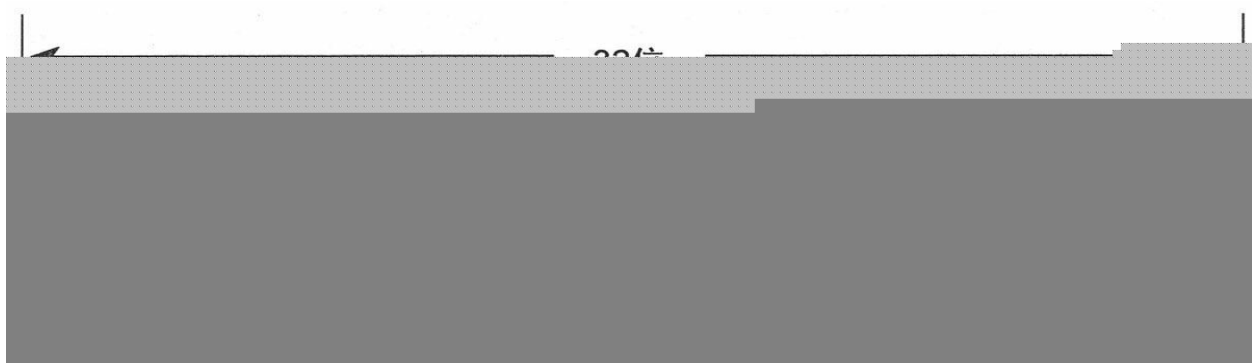


图8-56 一些子网地址能够被汇总成地址172.16.192.0/19。粗体字部分表明了每一个地址的网络位

路由器Whitney的配置参见示例8-106。

示例8-106 路由器Whitney带有地址汇总的OSPF配置

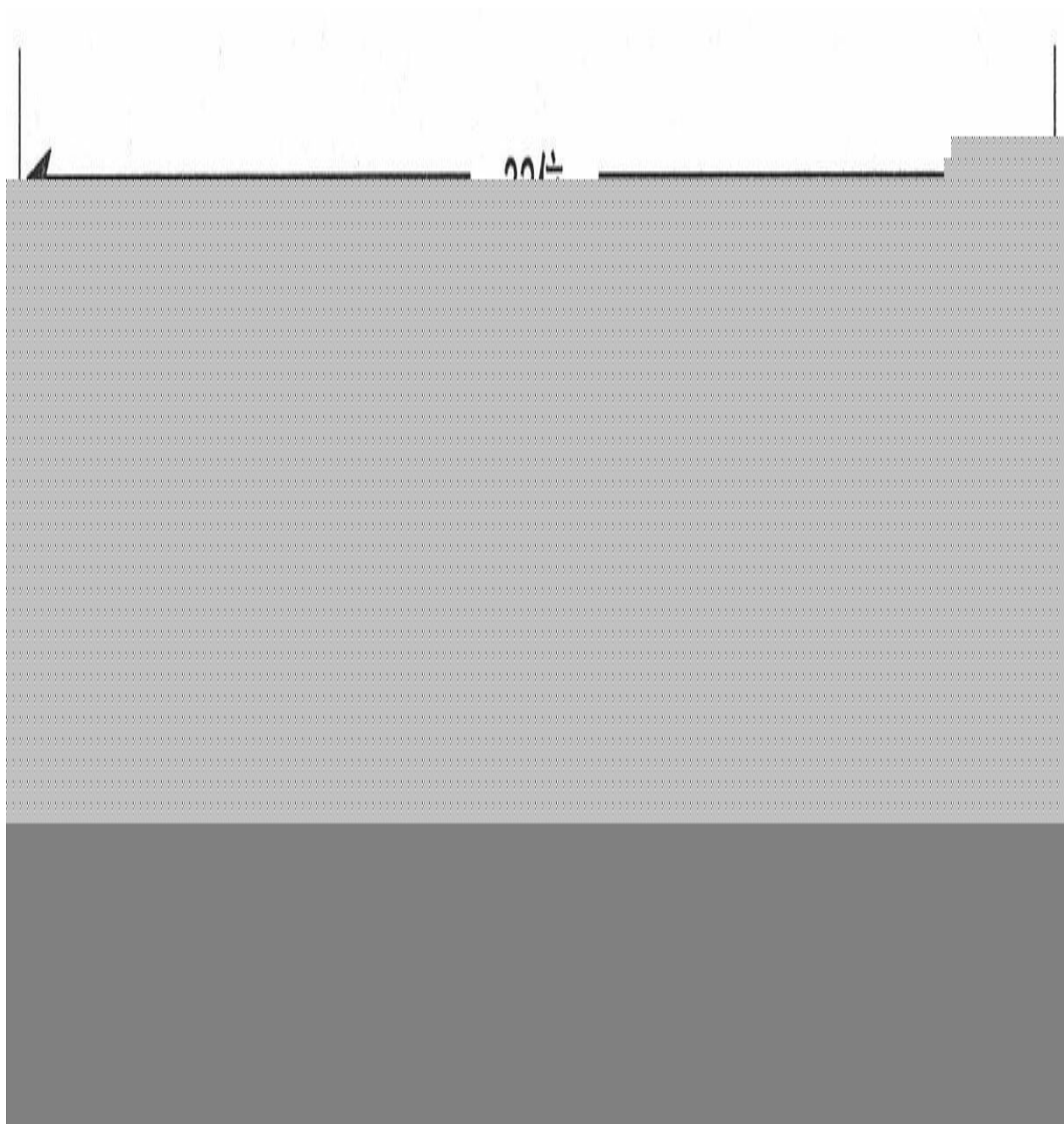


在其他3台ABR路由器上也做类似地配置。每台ABR路由器将向区域0通告与之相连的非骨干区域的汇总地址，并且也把区域0的子网地址汇总到了非骨干区域中去。

示例8-107中显示出了一个问题。在查看区域1的某台内部路由器的路由表时，发现区域0的子网地址没有被正确地汇总（讲得再清楚一点，区

域1的内部子网也没有显示出来)。虽然关于区域2和区域3的汇总地址被显示出来,但是在路由表中区域0的汇总地址却被它内部单独的子网替代了。

示例8-107 在区域1内的某台内部路由器的路由表中记录了区域0的个别子网,而替代了应该出现的汇总地址



当网络管理员以二进制的表示方式检查区域0的3个子网时,就会发现关

于区域0的**area range** 命令可能有问题（如图8-57所示）。

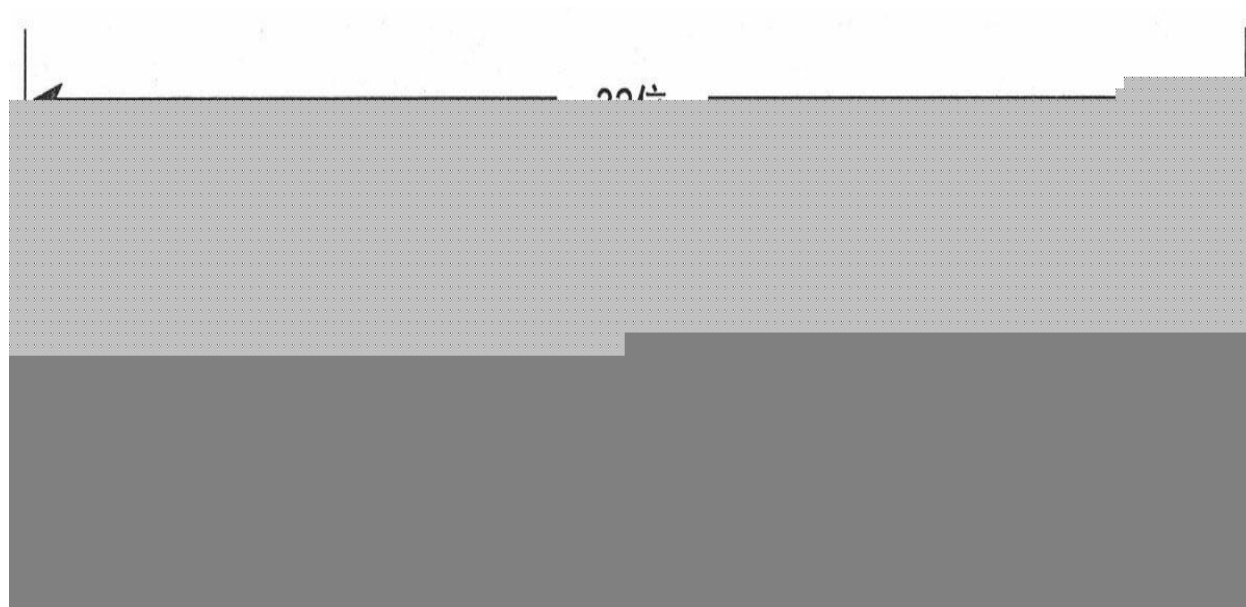
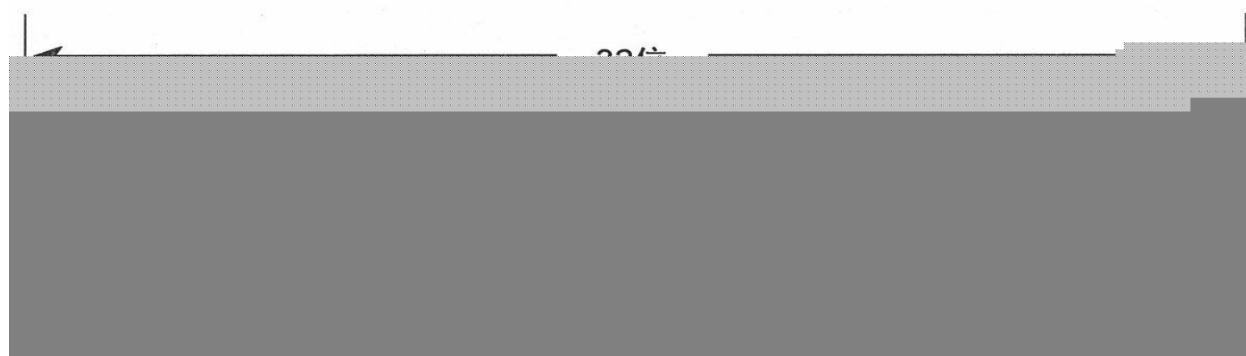


图8-57 区域0的子网、所配置的汇总掩码和正确的汇总地址

从上面可以看出，这里的问题是**area range** 命令指定汇总地址(172.16.113.0)比它携带的掩码(255.255.224.0)更具体了。对于这个19位的掩码，正确的地址应该是172.16.96.0（参见示例8-108）。

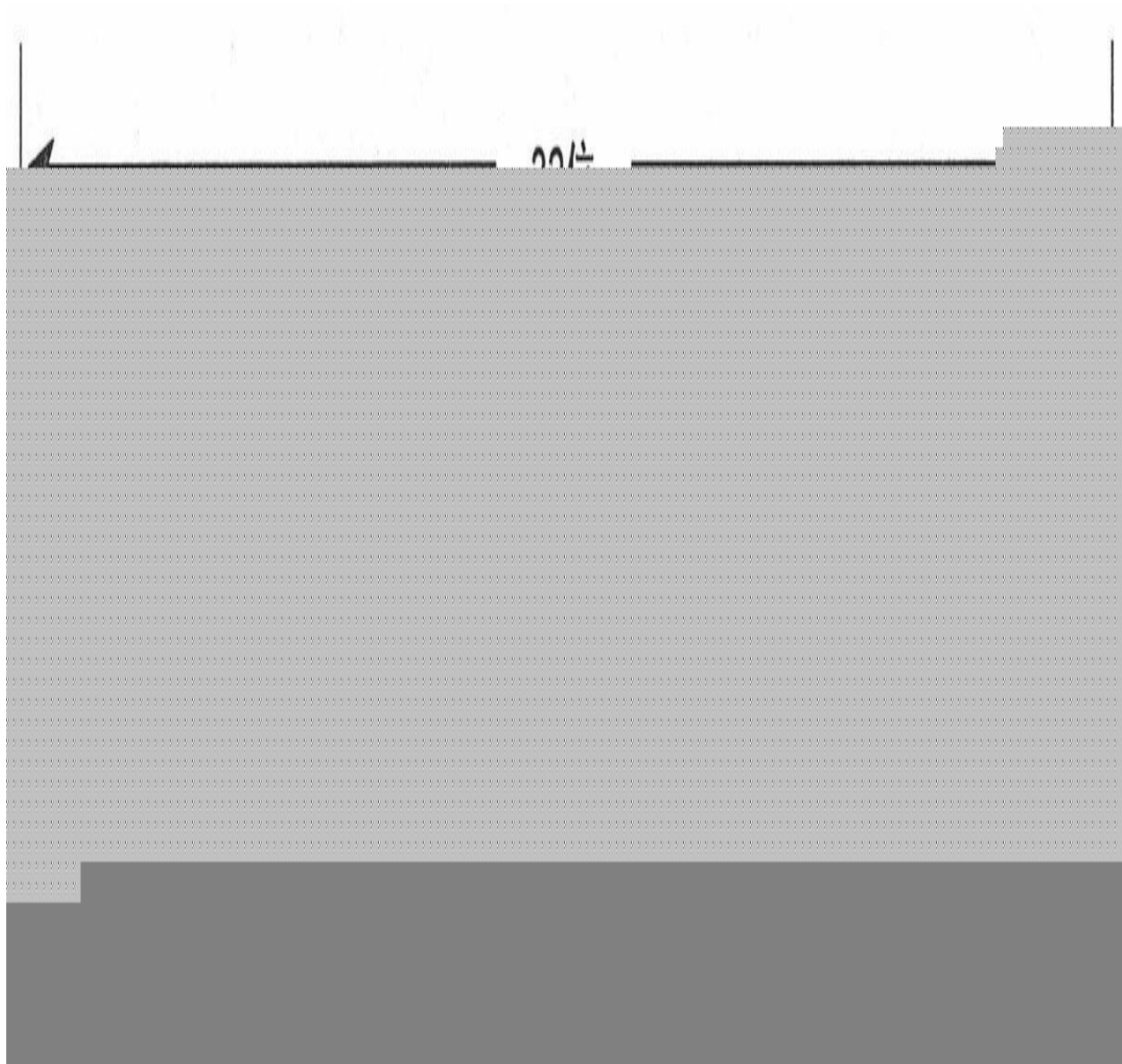
示例**8-108** 路由器**Whitney**带有正确的地址汇总的**OSPF**配置



示例8-109中显示了更改配置后的路由表的结果。当然，对于汇总区域0内的地址也有可以选择的其他地址。例如，172.16.113.0/24和172.16.113.0/27也都是可用的。最恰当的汇总地址是依赖于网络设计的优先性选择的。例如在示例8-101中显示的网络里，选用172.16.96.0/19是为了保持配置的一致性——所有的汇总地址都使用了19位的掩码。另

一方面，为了网络更好的扩展性，应该选用172.16.113.0/27作为汇总地址。在这个汇总地址里，还可以增加5个子网用于骨干区域，而剩余的更为广泛的地址范围可以用于网络上的其他地方。

示例8-109 区域0现在可以被正确地汇总了

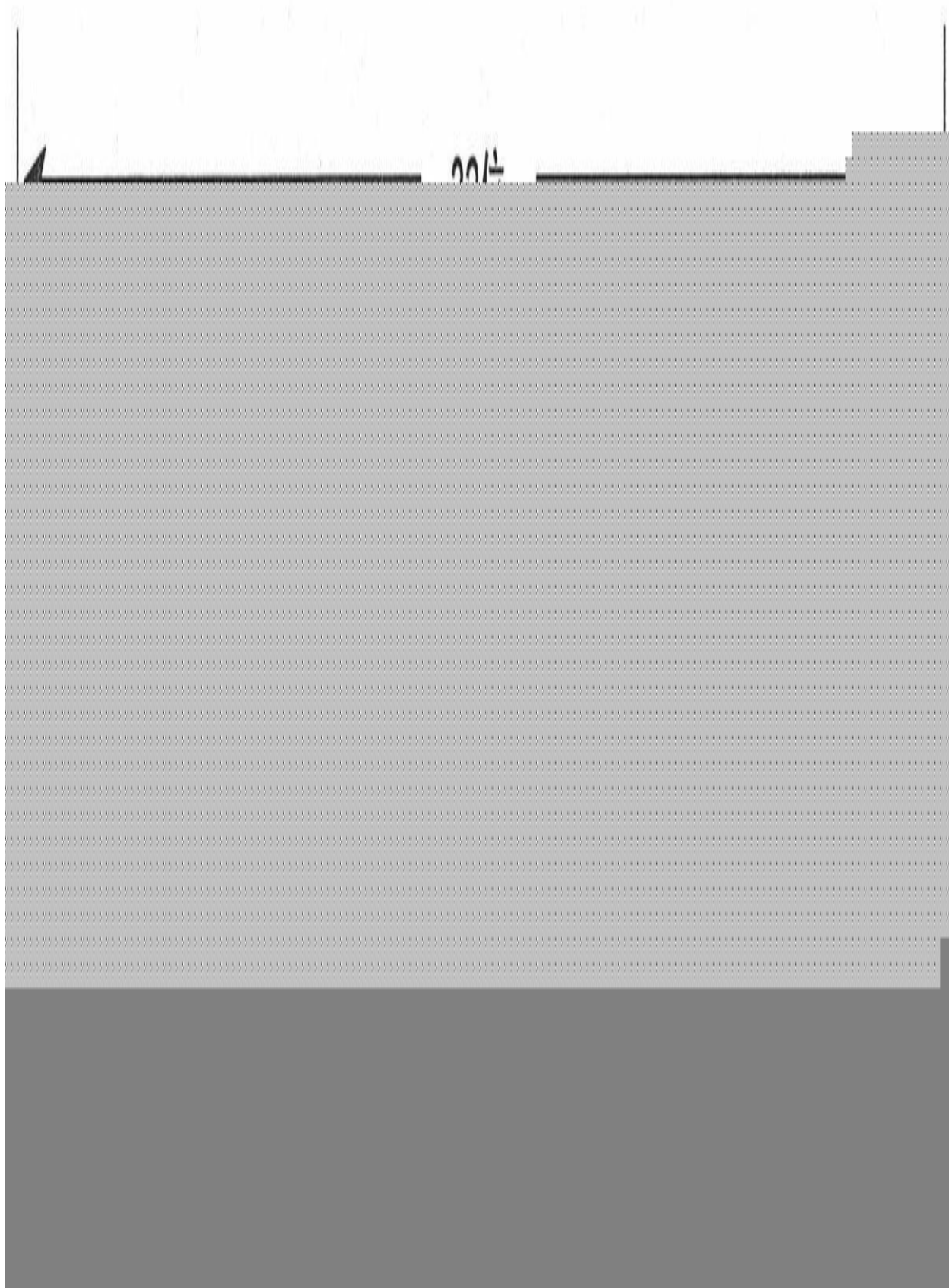


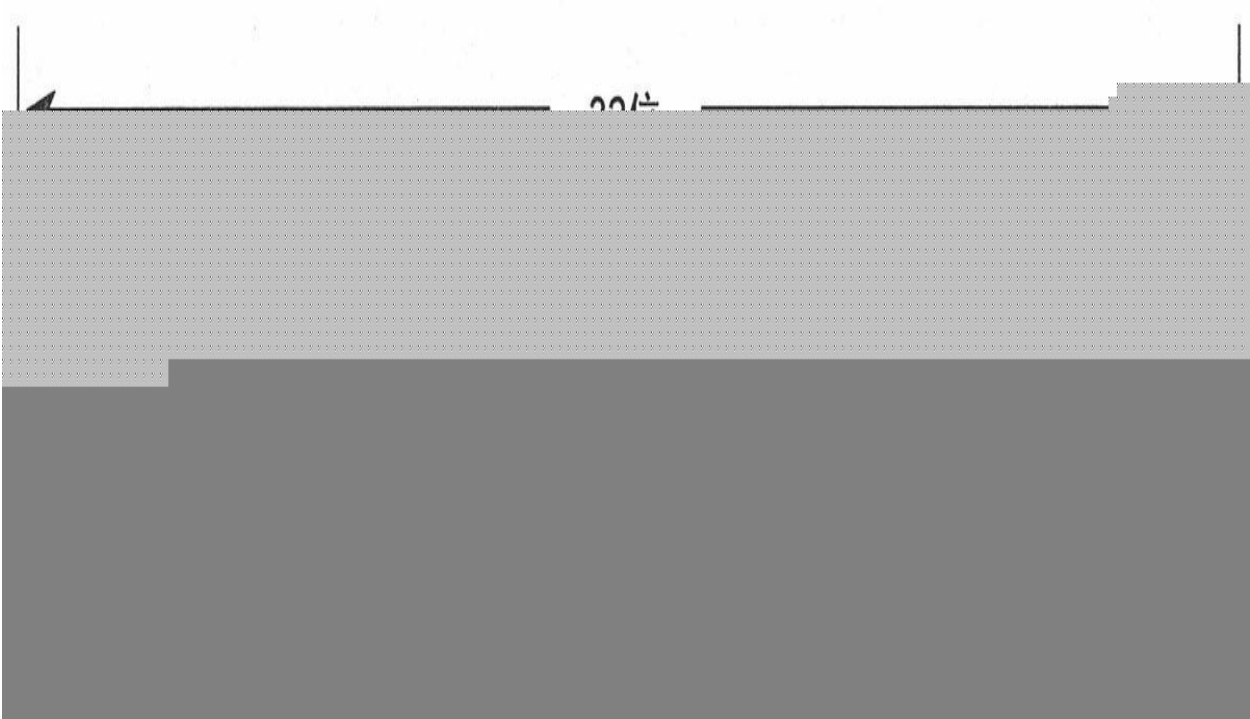
8.4 展 望

当提到链路状态路由选择协议的时候，大多数人们首先想到的是OSPF协议。但是，OSPF协议并不是IP网络上惟一的链路状态协议。ISO的中间系统-中间系统(IS-IS)虽然是设计用来为其他网络协议进行路由选择的，但是它也可以为IP网络进行路由选择。第10章将讲述这个鲜为人知的链路状态路由选择协议。

8.5 总结表：第8章命令总结

22位	





8.6 推荐读物

John Moy, “OSPF Version 2”, RFC 2328, 1998年4月。

John Moy, *OSPF: Anatomy of an Internet Routing Protocol* 。 Reading, Massachusetts:Addison-Wesley, 1998。

由OSPF的最初设计者和RFC的作者之一编写，这本书是非常值得阅读，它不仅全面阐述了OSPF协议，还给出了重要观点。第3章特别有趣，反映了在路由选择协议的设计、测试和标准化方面的内行观点。

8.7 复习题

1. 什么是OSPF邻居？
2. 什么是OSPF邻接关系？
3. OSPF数据包的5种类型是什么？每一种类型的用途是什么？
4. 什么是LSA?怎样区分一个LSA和一个OSPF更新数据包的不同？
5. LSA的类型1到类型5，以及类型7分别是什么？每一种类型的用途是什么？
6. 什么是链路状态数据库？链路状态数据库的同步是什么意思？
7. 什么是缺省的HelloInterval？
8. 什么是缺省的RouterDeadInterval？
9. 什么是路由器ID?怎样确定一个路由器ID？
10. 什么是区域？
11. 区域0的含义是什么？
12. 什么是最大生存时间(MaxAge)？
13. OSPF协议的4种路由器类型是什么？
14. OSPF协议的4种路径类型是什么？
15. OSPF协议的5种网络类型是什么？
16. 什么是指定路由器DR？
17. 在Cisco路由器上是怎样计算一个接口的出站代价的？
18. 什么是分段的区域？

19. 什么是虚链路？
20. 末梢区域、完全末梢区域和非纯末梢区域之间有什么不同？
21. OSPF网络条目和OSPF路由器条目之间有什么不同之处？
22. 为什么类型2的认证方式比类型1的认证方式更好？
23. 在LSA头部中哪3个字段是用来区分不同的LSA的？另外，在LSA头部中哪3个字段是用来区分相同LSA的不同实例的？

8.8 配置练习

1. 表8-13显示了14台路由器的接口和地址，其中也表明了每一个接口相连的OSPF区域。根据表中提供的信息，假定下面的事实：

- 每一台路由器的所有接口都显示在表中。
- 如果没有区域显示(-)，就表示在相关的接口上不运行OSPF协议。
- 子网地址的第二个八位组字节和区域ID相同。
- 每一个OSPF接口地址的前16位指定一个区域。例如，前缀是10.30.x.x的地址将只能在区域30中出现。

请写出表8-13中的路由器关于OSPF的配置（提示：可以首先画出一台路由器和子网的拓扑图）。

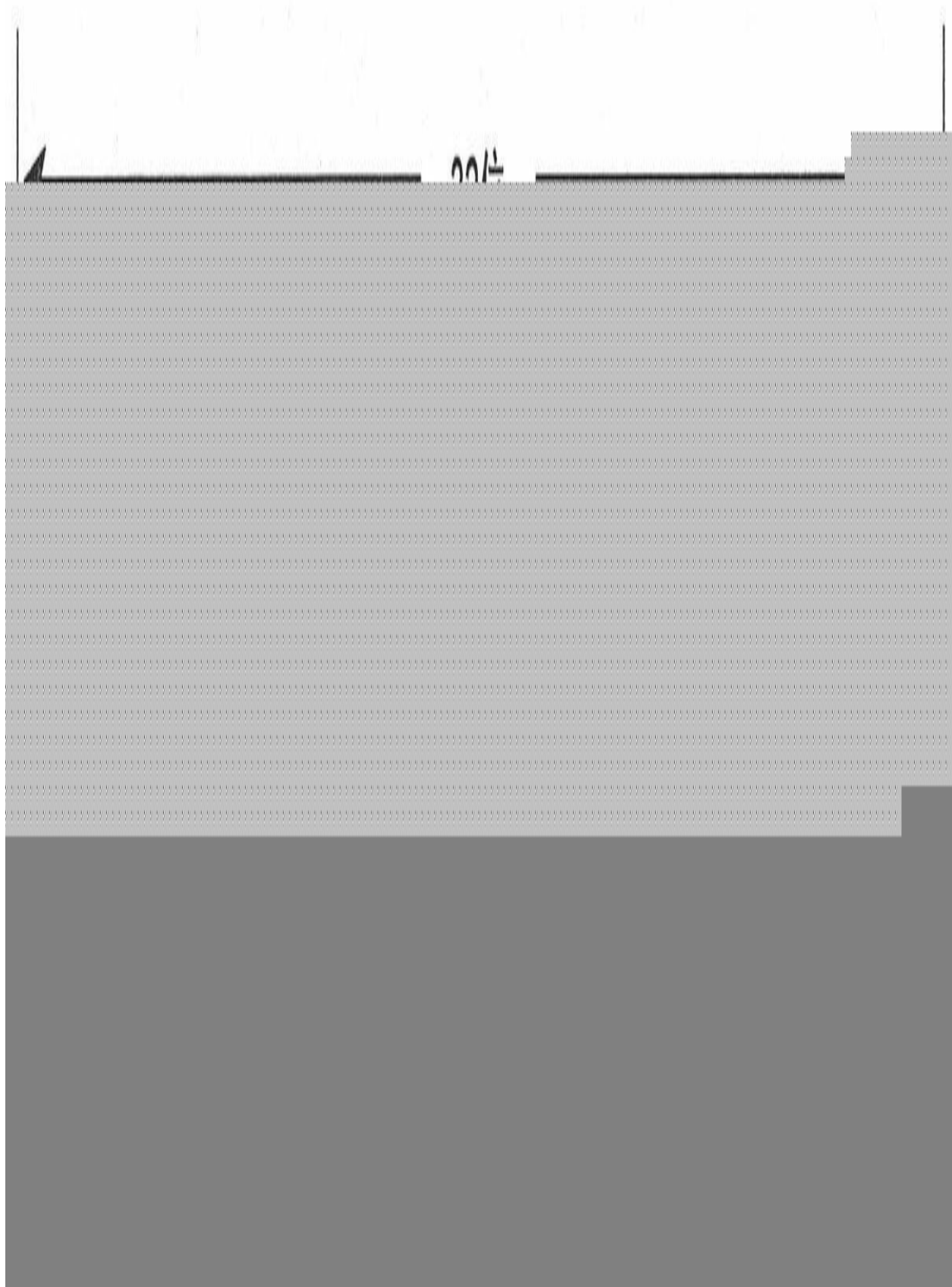


表8-13

关于配置练习1~6的路由器信息

路由器		接口	IP地址	子网掩码	OSPF区域
R1	R1-0/0/0	GE0/0/0	10.0.0.1	255.255.255.0	0
		GE0/0/1	10.0.0.2	255.255.255.0	0
R2	R2-0/0/0	GE0/0/0	10.0.0.3	255.255.255.0	0
		GE0/0/1	10.0.0.4	255.255.255.0	0
R3	R3-0/0/0	GE0/0/0	10.0.0.5	255.255.255.0	0
		GE0/0/1	10.0.0.6	255.255.255.0	0
R4	R4-0/0/0	GE0/0/0	10.0.0.7	255.255.255.0	0
		GE0/0/1	10.0.0.8	255.255.255.0	0
R5	R5-0/0/0	GE0/0/0	10.0.0.9	255.255.255.0	0
		GE0/0/1	10.0.0.10	255.255.255.0	0
R6	R6-0/0/0	GE0/0/0	10.0.0.11	255.255.255.0	0
		GE0/0/1	10.0.0.12	255.255.255.0	0
R7	R7-0/0/0	GE0/0/0	10.0.0.13	255.255.255.0	0
		GE0/0/1	10.0.0.14	255.255.255.0	0
R8	R8-0/0/0	GE0/0/0	10.0.0.15	255.255.255.0	0
		GE0/0/1	10.0.0.16	255.255.255.0	0
R9	R9-0/0/0	GE0/0/0	10.0.0.17	255.255.255.0	0
		GE0/0/1	10.0.0.18	255.255.255.0	0
R10	R10-0/0/0	GE0/0/0	10.0.0.19	255.255.255.0	0
		GE0/0/1	10.0.0.20	255.255.255.0	0
R11	R11-0/0/0	GE0/0/0	10.0.0.21	255.255.255.0	0
		GE0/0/1	10.0.0.22	255.255.255.0	0
R12	R12-0/0/0	GE0/0/0	10.0.0.23	255.255.255.0	0
		GE0/0/1	10.0.0.24	255.255.255.0	0
R13	R13-0/0/0	GE0/0/0	10.0.0.25	255.255.255.0	0
		GE0/0/1	10.0.0.26	255.255.255.0	0
R14	R14-0/0/0	GE0/0/0	10.0.0.27	255.255.255.0	0
		GE0/0/1	10.0.0.28	255.255.255.0	0
R15	R15-0/0/0	GE0/0/0	10.0.0.29	255.255.255.0	0
		GE0/0/1	10.0.0.30	255.255.255.0	0
R16	R16-0/0/0	GE0/0/0	10.0.0.31	255.255.255.0	0
		GE0/0/1	10.0.0.32	255.255.255.0	0
R17	R17-0/0/0	GE0/0/0	10.0.0.33	255.255.255.0	0
		GE0/0/1	10.0.0.34	255.255.255.0	0
R18	R18-0/0/0	GE0/0/0	10.0.0.35	255.255.255.0	0
		GE0/0/1	10.0.0.36	255.255.255.0	0
R19	R19-0/0/0	GE0/0/0	10.0.0.37	255.255.255.0	0
		GE0/0/1	10.0.0.38	255.255.255.0	0
R20	R20-0/0/0	GE0/0/0	10.0.0.39	255.255.255.0	0
		GE0/0/1	10.0.0.40	255.255.255.0	0
R21	R21-0/0/0	GE0/0/0	10.0.0.41	255.255.255.0	0
		GE0/0/1	10.0.0.42	255.255.255.0	0
R22	R22-0/0/0	GE0/0/0	10.0.0.43	255.255.255.0	0
		GE0/0/1	10.0.0.44	255.255.255.0	0
R23	R23-0/0/0	GE0/0/0	10.0.0.45	255.255.255.0	0
		GE0/0/1	10.0.0.46	255.255.255.0	0
R24	R24-0/0/0	GE0/0/0	10.0.0.47	255.255.255.0	0
		GE0/0/1	10.0.0.48	255.255.255.0	0
R25	R25-0/0/0	GE0/0/0	10.0.0.49	255.255.255.0	0
		GE0/0/1	10.0.0.50	255.255.255.0	0
R26	R26-0/0/0	GE0/0/0	10.0.0.51	255.255.255.0	0
		GE0/0/1	10.0.0.52	255.255.255.0	0
R27	R27-0/0/0	GE0/0/0	10.0.0.53	255.255.255.0	0
		GE0/0/1	10.0.0.54	255.255.255.0	0
R28	R28-0/0/0	GE0/0/0	10.0.0.55	255.255.255.0	0
		GE0/0/1	10.0.0.56	255.255.255.0	0
R29	R29-0/0/0	GE0/0/0	10.0.0.57	255.255.255.0	0
		GE0/0/1	10.0.0.58	255.255.255.0	0
R30	R30-0/0/0	GE0/0/0	10.0.0.59	255.255.255.0	0
		GE0/0/1	10.0.0.60	255.255.255.0	0
R31	R31-0/0/0	GE0/0/0	10.0.0.61	255.255.255.0	0
		GE0/0/1	10.0.0.62	255.255.255.0	0
R32	R32-0/0/0	GE0/0/0	10.0.0.63	255.255.255.0	0
		GE0/0/1	10.0.0.64	255.255.255.0	0
R33	R33-0/0/0	GE0/0/0	10.0.0.65	255.255.255.0	0
		GE0/0/1	10.0.0.66	255.255.255.0	0
R34	R34-0/0/0	GE0/0/0	10.0.0.67	255.255.255.0	0
		GE0/0/1	10.0.0.68	255.255.255.0	0
R35	R35-0/0/0	GE0/0/0	10.0.0.69	255.255.255.0	0
		GE0/0/1	10.0.0.70	255.255.255.0	0
R36	R36-0/0/0	GE0/0/0	10.0.0.71	255.255.255.0	0
		GE0/0/1	10.0.0.72	255.255.255.0	0
R37	R37-0/0/0	GE0/0/0	10.0.0.73	255.255.255.0	0
		GE0/0/1	10.0.0.74	255.255.255.0	0
R38	R38-0/0/0	GE0/0/0	10.0.0.75	255.255.255.0	0
		GE0/0/1	10.0.0.76	255.255.255.0	0
R39	R39-0/0/0	GE0/0/0	10.0.0.77	255.255.255.0	0
		GE0/0/1	10.0.0.78	255.255.255.0	0
R40	R40-0/0/0	GE0/0/0	10.0.0.79	255.255.255.0	0
		GE0/0/1	10.0.0.80	255.255.255.0	0
R41	R41-0/0/0	GE0/0/0	10.0.0.81	255.255.255.0	0
		GE0/0/1	10.0.0.82	255.255.255.0	0
R42	R42-0/0/0	GE0/0/0	10.0.0.83	255.255.255.0	0
		GE0/0/1	10.0.0.84	255.255.255.0	0
R43	R43-0/0/0	GE0/0/0	10.0.0.85	255.255.255.0	0
		GE0/0/1	10.0.0.86	255.255.255.0	0
R44	R44-0/0/0	GE0/0/0	10.0.0.87	255.255.255.0	0
		GE0/0/1	10.0.0.88	255.255.255.0	0
R45	R45-0/0/0	GE0/0/0	10.0.0.89	255.255.255.0	0
		GE0/0/1	10.0.0.90	255.255.255.0	0
R46	R46-0/0/0	GE0/0/0	10.0.0.91	255.255.255.0	0
		GE0/0/1	10.0.0.92	255.255.255.0	0
R47	R47-0/0/0	GE0/0/0	10.0.0.93	255.255.255.0	0
		GE0/0/1	10.0.0.94	255.255.255.0	0
R48	R48-0/0/0	GE0/0/0	10.0.0.95	255.255.255.0	0
		GE0/0/1	10.0.0.96	255.255.255.0	0
R49	R49-0/0/0	GE0/0/0	10.0.0.97	255.255.255.0	0
		GE0/0/1	10.0.0.98	255.255.255.0	0
R50	R50-0/0/0	GE0/0/0	10.0.0.99	255.255.255.0	0
		GE0/0/1	10.0.0.100	255.255.255.0	0

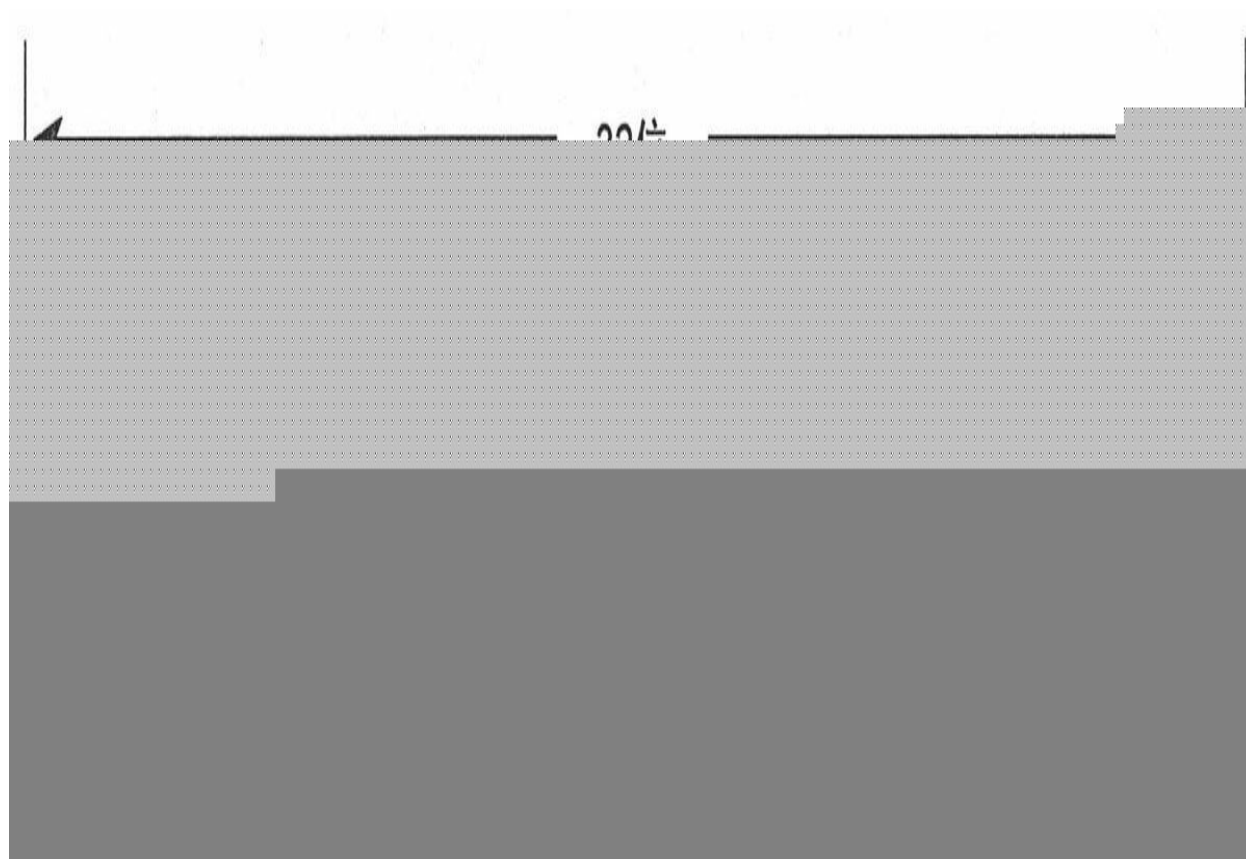
2. 在表8-13中的所有ABR路由器上配置路由汇总。
3. 更改配置，使区域15成为一个末梢区域。
4. 更改配置，使区域30成为一个完全末梢区域。
5. 路由器H的SO接口和一个运行其他路由选择协议的路由器相连，并将从该路由选择协议学习到的路由重新分配到OSPF域中。更改必要的配置，以便使这些重新分配的路由可以在整个OSPF域内通告，但是不允许任何类型5的LSA通告到区域20中。

6. 在路由器C和路由器M之间的串行链路是一条带宽很低的链路。更改配置，以便使OSPF协议把这条链路作为一个按需链路看待。

8.9 故障排除练习

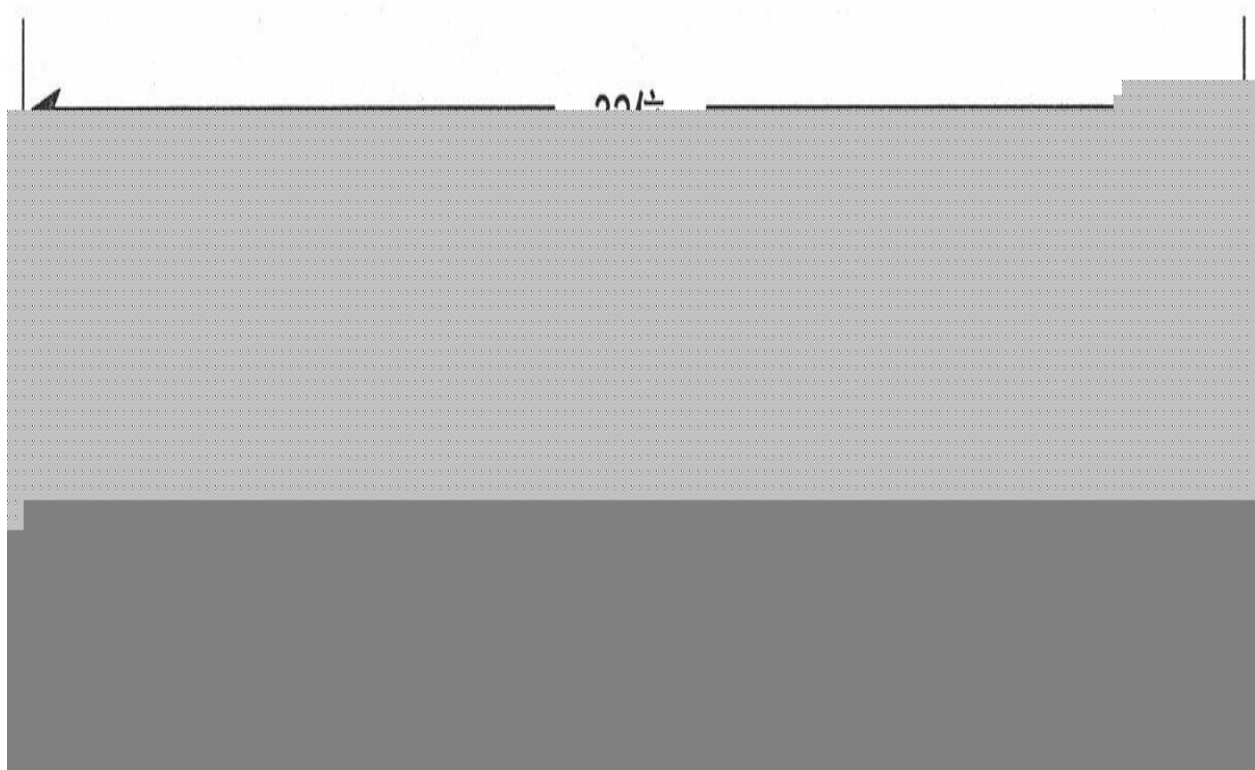
1. 在两台路由器上的OSPF不能工作。当打开debug命令进行调试后，发现每10s就会收到示例8-110中显示的信息。请问网络发生了什么故障？

示例8-110 故障排除练习1的调试信息



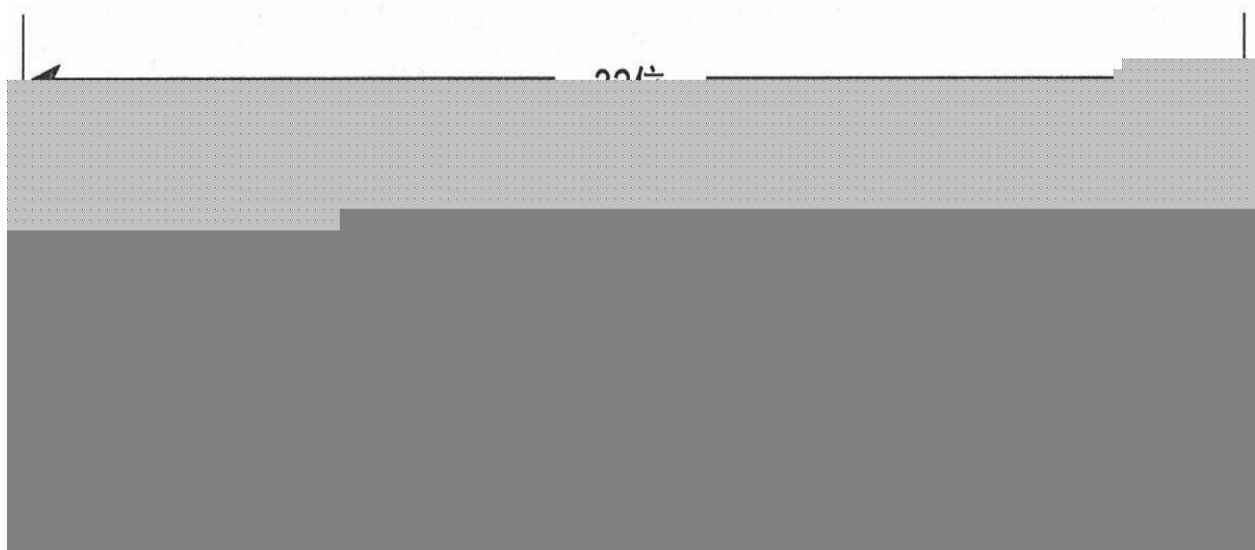
2. 根据示例8-111中显示的调试信息，查明网络出现的故障。

示例8-111 故障排除练习2的调试信息



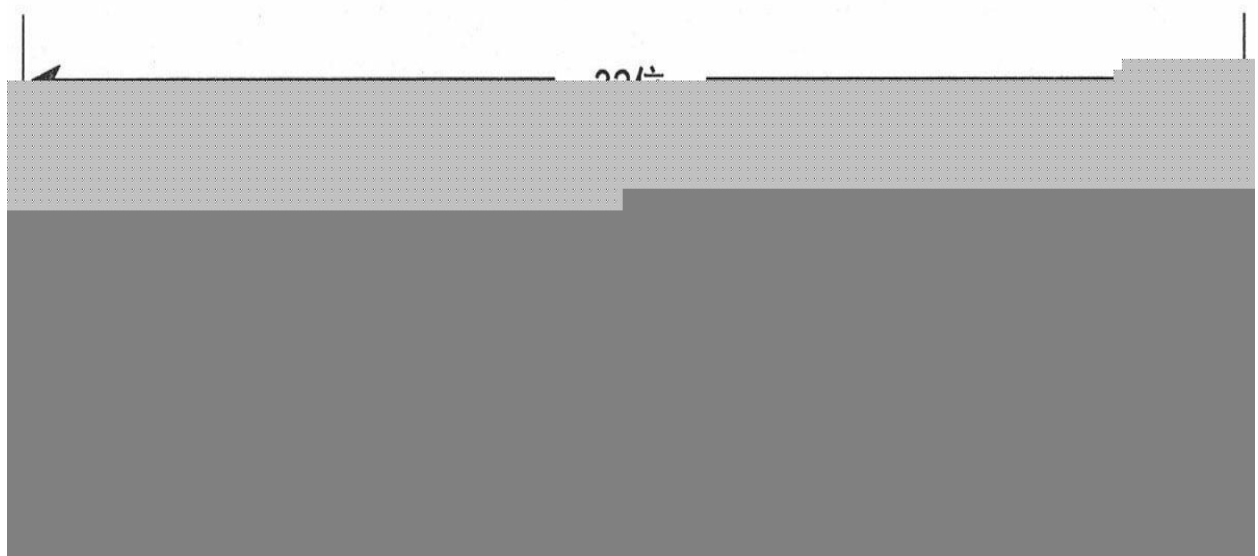
3. 根据示例8-112中显示的错误信息，查明网络出现的故障。

示例8-112 故障排除练习3的调试信息



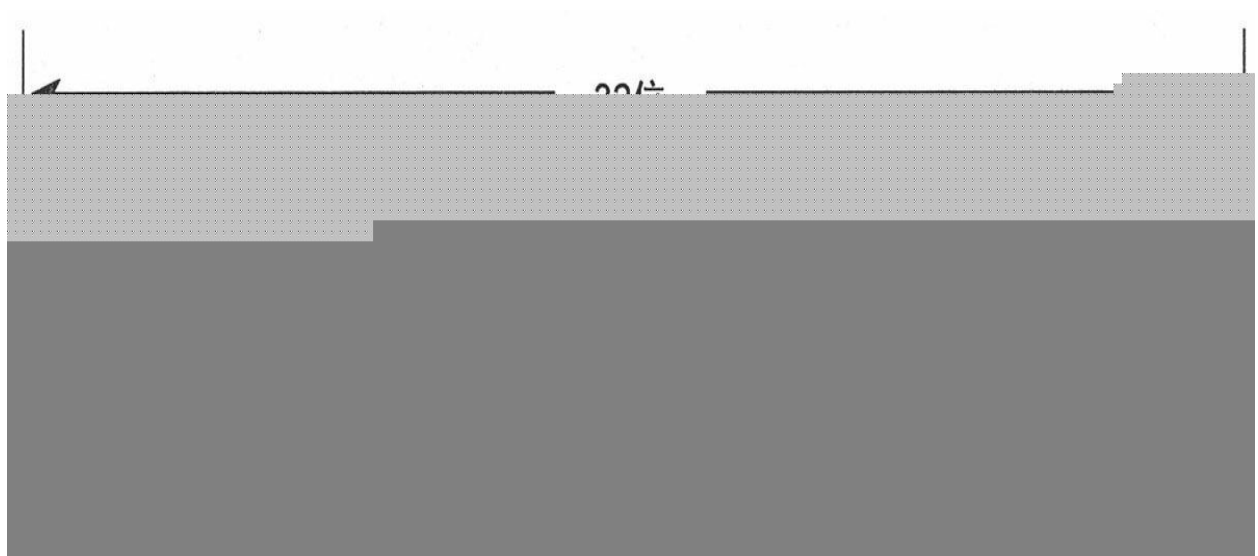
4. 根据示例8-113中显示的错误信息，查明网络出现的故障。

示例8-113 故障排除练习4的调试信息



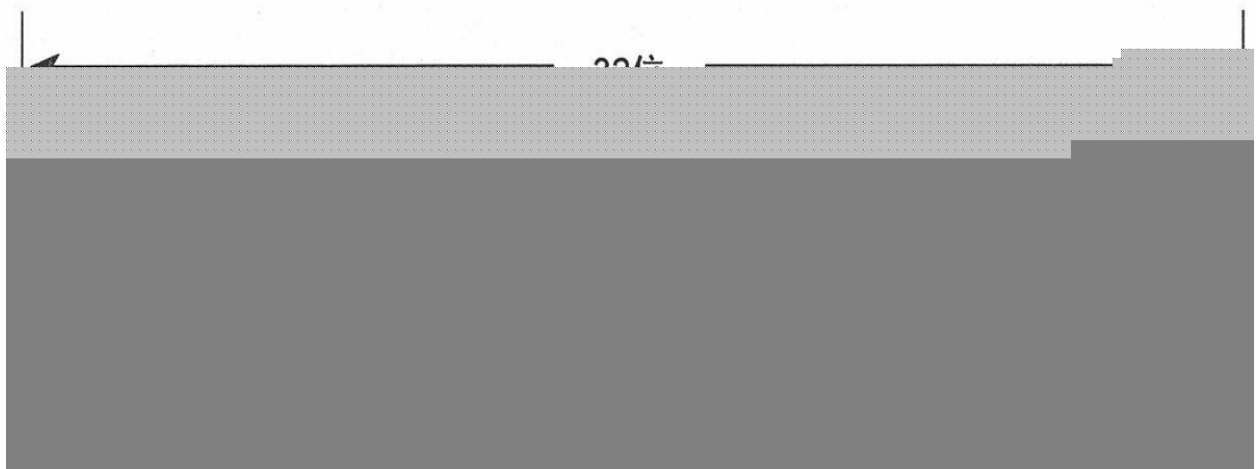
5. 根据示例8-114中显示的错误信息，查明网络出现的故障。

示例**8-114** 故障排除练习5的调试信息

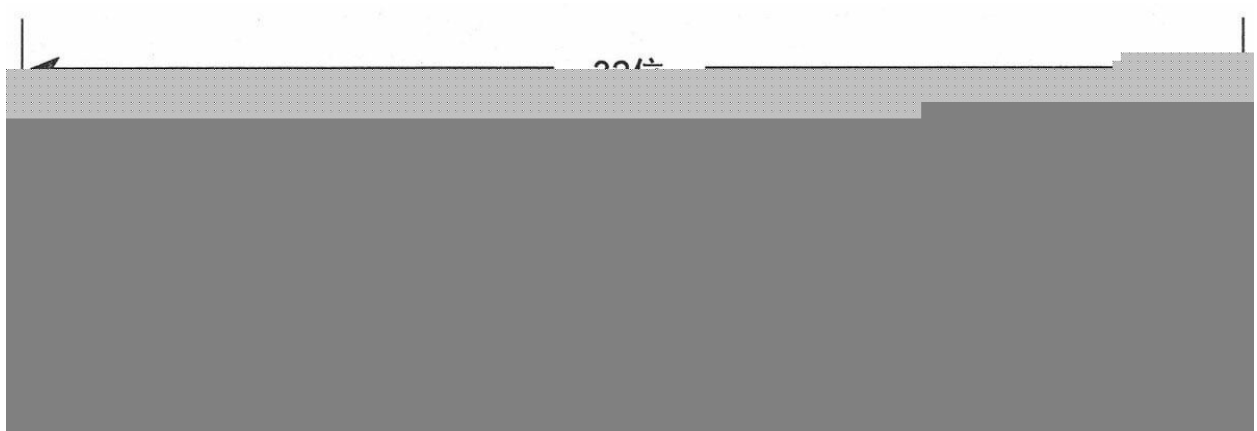


6. 在图8-58中所示的路由器上做如下配置。

路由器A:



路由器B:



在这里，路由器A和B不能形成一个邻接关系。请问发生了什么问题？

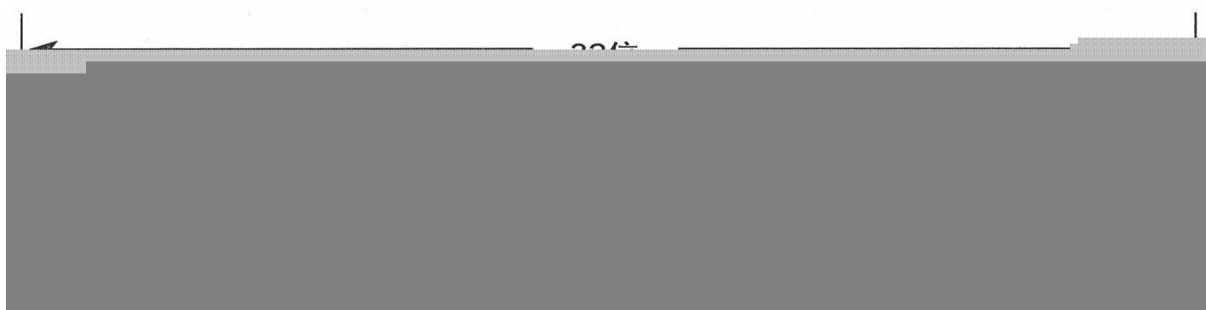
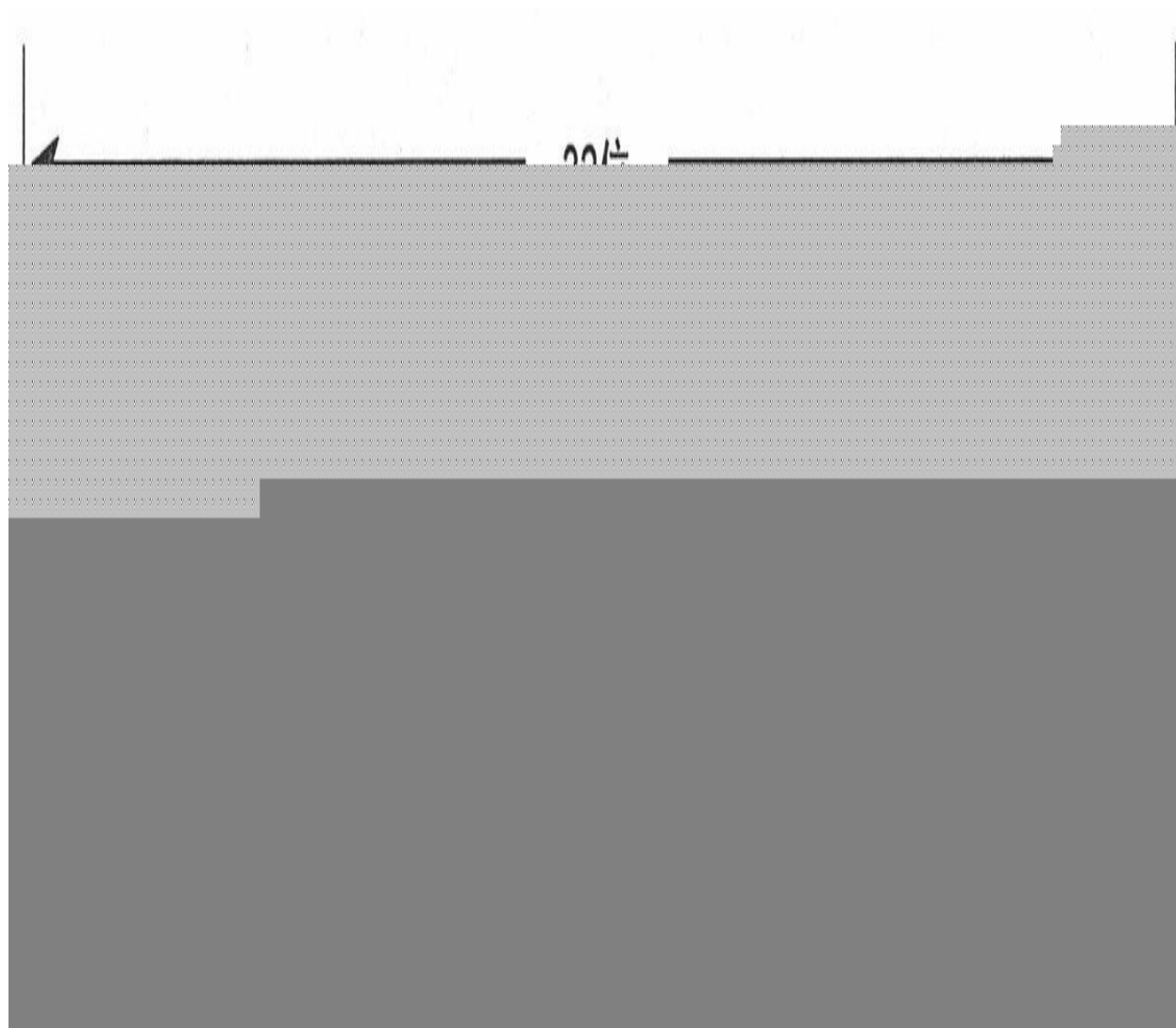


图8-58 故障排除练习6的调试信息

7. 示例8-115显示了某个区域里的一个链路状态数据库，其中这个区域里存在一条不稳定的链路。基于所显示的信息，哪一条链路看起来像是

有问题的链路？

示例8-115 故障排除练习7的链路状态数据库



[1] 更准确地说，RFC建议称为等价多路径——多条等价路径的发现和使用，但是RFC并没有指明路由选择协议如何路由转发单个数据包通过这些多条不同的路径。在Cisco路由器上，OSPF的实现执行的是前面章节描述过的等价负载均衡。

[2] 由于OSPF协议的术语和概念的相互关系，本章在完整地描述它们的定义之前将会频繁地使用这些术语和概念。建议读者多阅读几遍本章的内容，而不是仅仅阅读一遍，以便确保对OSPF协议的操作有一个完全

的理解。复习第4章中的4.3节也是很有帮助的。

[3] 受到路由过滤影响的是利用链路状态数据库计算路由的基本过程，而不是邻居之间交换路由的基本过程。如需了解更详细的内容，请参见第13章的内容。

[4] 在RFC 2328中没有为HelloInterval或者RouterDeadInterval设置一个必需的值，虽然它建议采用10s和4xHelloInterval。

[5] 这个规则的一个例外是重传的LSA数据包，它们在所有的网络类型中都是使用单播方式发送的。这个例外的情况将在后面的“可靠的泛洪扩散：确认”部分讲述。

[6] 注意，不要把stub网络和本章后面章节讲述的stub区域的概念相混淆。

[7] 从IOS 11.3版本开始，这个缺省的行为可以在loopback接口上增加使用命令`ip ospf network point-to-point`来改变。这将可以使loopback接口的地址作为一个子网地址来通告。

[8] 根据所运行的IOS软件版本而定，该命令输出的信息可能比这里讨论的更多，但这里的信息是每一个OSPF接口最基本的信息。

[9] 请注意，在这里和这个接口相连的是帧中继网络。但是由于这是一个点到点的子接口，因此，OSPF协议使用点到点类型替代了NBMA的网络类型。

[10] 请与示例8-1中的用法进行比较；看看有什么不同。

[11] 当你使用命令`show ip ospf interface`时，根据你所运行的IOS软件版本而定，你可能会看到某些与示例8-6中显示的不同的信息。

[12] Alex McKenzie, “ISO Transport Protocol Specification ISO DP 8073, ”RFC 905, April 1984, Annex B.

[13] 在IOS11.3AA版本中引入了组步调命令`timers lsa-group-pacing`;而从IOS12.2 (4) T版本开始命令改变为`timers pacing lsa-group`。

[14] 这个距离矢量的行为就是OSPF协议为什么需要一个骨干区域和为什么要求所有的域间通信量都必须通过骨干区域的原因。通过把这些区域构成一个本质上像星型一样的网络拓扑，可以避免距离矢量协议中易于出现的路由环路。

[15] 缺省路由是指在路由表中，如果没有更具体的路由存在就可以选用的路由。OSPF协议和RIP协议使用IP地址0.0.0.0来表示一个缺省路由。更详细的信息请参考第12章的内容。

[16] John Moy, “Multicast Extensions to OSPF,” RFC 1584, 1994年3月。

[17] Rob Coltun, “The OSPF Opaque LSA Option,” RFC 2370, 1998年7月。

[18] Rob Coltun and Vince Fuller, “The OSPF NSSA Option,” RFC 1587, 1994年3月。

[19] 这里描述的路由表查找过程是和RFC 2328一致的。早期的OSPF RFC规定首先要创建一个匹配路由的集合，然后再选择优先的路径类型，最后再进行最长匹配。

[20] 在写这个版本的书时，Cisco IOS软件已经支持通过命令maximum-paths将最大值设为6~16。

[21] John Moy, “Extending OSPF to Support Demand Circuits,” RFC 1793, 1995年4月。

[22] 虽然按需电路上的OSPF可能被配置在任何接口上，但是，在多路访问类型的网络上不能抑制Hello数据包的发送——如果这么做会阻碍DR处理的一些相应功能。结果是，这个增强特性仅仅在点到点和点到多点类型的网络上才有实际用途。

[23] 注意，这意味着MaxAge实际上是MaxAge+DoNotAge。

[24] Philip Almquist, “Type of Service in the Internet Protocol Suite,” RFC 1349, 1992年7月。

[25] 虽然可以在同一台路由器上配置多个OSPF进程，但是并不十分鼓励

这么做，因为，运行多个数据库将要占用较多的路由器资源。

[26] 请参考附录B的内容，查找有关于反向掩码的用法说明。

[27] 如果仅仅是断开接口连接并不会引起路由器ID的改变。

[28] N2表明的是和E2相同的度量计算——也就是说，只使用外部的代价。随后的一个例子将演示E1和N1的度量类型。

[29] 注意，在**redistribution** 命令中使用了*metric-type 1* 参数。这个参数的作用是使外部的目的路由利用E1度量来通告。在NSSA中，度量类型变成了N1,参见示例8-53所示。

[30] 增加这条路由的理由将在第11章和第12章中结合实例，作更详细的讲述。

[31] 当一条虚链路穿过一个非骨干区域连接某个区域到骨干区域时，其中一台ABR路由器将位于这两个非骨干区域之间。

[32] 根据你使用的IOS版本的不同，命令**show ip ospf virtual-link** 的输出可能有点不同。

[33] 具体来说，就是ospfExternLSACksumSum和ospfAreaLSACksumSum。这些是单独的LSA校验和字段的总和。由于校验和的计算包括序列号，并且序列号可能不同，因此，校验和也可能不同。

[34] 也就是ospfExternLsaCksumSum和ospfAreaLsaCksumSum。

本章包括以下主题：

- OSPFv3的基本原理与实现；
- OSPFv3的配置；
- OSPFv3故障诊断。

第9章

OSPFv3

正如读者在前面章节中所看到的，IPv6的路由选择需要对协议进行一些改动。首先最主要的改动就是，每个协议消息都必须改为能够传送长度为IPv4地址4倍的地址。理论上来说，这些可以通过开放最短路径优先(OSPF)协议来实现，要么通过更改现有的链路状态通告(LSA)，要么定义新的LSA。但是，OSPF协议的发展早在20世纪80年代末就开始了，当时路由器的性能很低，时延很大，而且内存昂贵。现在这些问题都不存在了，OSPFv2的一些用来适应和调节那些早期连网实际情况的特性现在已不实用了。更进一步来说，OSPFv2协议的大量实际应用经验表明，它在某些领域显得没有效率。

因此，在起初考虑扩展OSPF支持IPv6的时候，就意识到这是一个改进优化OSPF协议本身的机会。结果是，不仅仅为IPv6对OSPFv2进行了扩展，还创建了一个新的OSPF的改进版本——OSPF第3版。

9.1 OSPFv3的基本原理与实现

OSPFv3在RFC 2740中有详细描述。OSPFv3与OSPFv2的关系，非常类似于RIPng与RIPv2的关系。最重要的是，OSPFv3使用了与OSPFv2相同的基本实现机制——SPF算法、泛洪扩散、DR选举、区域等。还有一些像计时器与度量等常量和变量也是相同的。

另外一个和RIPng与RIPv2的关系的类似之处是，OSPFv3也不向后兼容OSPFv2。因此，如果读者希望在IPv4和IPv6环境中同时使用OSPF协议，就必须同时运行OSPFv2和OSPFv3协议。在撰写本章的时候，有一个有关增加IPv4协议支持OSPFv3协议的讨论，但是还没有完成相关的细节和规范。作为个人来讲，我希望增加这个支持，因为OSPFv3协议是一个重要的改进协议。

在这里，假定读者已经阅读了前面的章节，并掌握了OSPFv2的有关基本原理与实现。本章将不再重复讲述那些内容，而仅仅阐明OSPFv3的重要不同之处——主要的基本原理与LSA格式。

9.1.1 OSPFv3与OSPFv2的不同之处

除了在下面章节讲述的有关LSA的变化外，还有一些OSPF机制本身的变化。在本小节中将讲述其中一些最重要的变化。

- 每个链路上的协议处理 ——读者在前面的章节中已经看到，一条链路的接口可以拥有多个IPv6的地址。事实上，单条链路可以属于多个子网，与同一条链路相连但属于不同的IPv6子网的两个接口仍然可以通信。在OSPFv3中将OSPFv2的“子网”概念改变为了“链路”概念，而且允许在同一条链路上但属于不同IPv6子网的两个邻居交换数据包。
- 取消了寻址概念 ——正如读者在随后的章节中所读到的，OSPFv3的路由器LSA和网络LSA都不再携带IP地址。这里会定义一个新的LSA来实现这项功能，这具有一些扩展性方面的好处。但是，32位的RID、AID，以及LSA ID都保留在IPv6中。记住，虽然这些ID是用点分十进制编写的，经常来自工作在IPv4地址环境中的OSPFv2网络，但是它们不是IPv4地址。这些OSPFv3的ID仍然用点

分十进制表示，允许在现有的OSPFv2网络上简单地叠加一个OSPFv3网络。

- 邻居总是通过路由器**ID**来标识 ——在广播网络和NBMA网络的链路上，OSPFv2邻居是通过它们的接口地址来标识的，而其他类型链路上的邻居是通过RID来标识的。OSPFv3取消了这种不一致性，在所有类型的链路上的所有邻居都通过RID来标识。
- 增加了链路本地泛洪扩散范围 ——OSPFv3保留了OSPFv2中域（或AS）和区域(area)泛洪扩散的范围，但增加了一个链路本地泛洪扩散的范围。读者从第2章的介绍中已经了解到，IPv6使链路本地范围具有很多用途。正如读者将在后面章节所看到的，增加新的LSA——链路LSA (Link LSA)用来携带仅仅与单个链路上的邻居相关联的信息。这种LSA具有链路本地泛洪扩散的范围，这就意味着它不能超出任何相连的路由器的范围而进行扩散。
- 链路本地地址的使用 ——读者已经知道OSPFv2的数据包具有链路本地的范围，它们不能被任何路由器转发。OSPFv3则使用路由器的链路本地IPv6地址（回忆在第2章中，这些地址总是以FF80::/10开头的）作为源地址和下一跳地址。
- 对每个链路上多个实例的支持 ——这种支持应用在台OSPF路由器连接到单个广播链路上，但它们又不属于单个邻接关系的情况。这种情况的一个例子就是共享的网络接入点（Network Access Point, NAP）。例如，假定有4台路由器连接到同一条以太网链路。路由器1和路由器2属于一个OSPF域，路由器3和路由器4属于另外一个不同的OSPF域。那么，它们应该分别在路由器1和路由器2之间以及路由器3和路由器4之间建立相应的邻接关系，而不是建立在路由器1和路由器3之间。读者可以利用认证操作来完成这种OSPFv2邻接关系的隔离，但这不是一种理想的方法。在另外一些情况，属于一个邻接关系的路由器将会不断地记录被其他邻接拒绝的Hello包的认证失败记录。OSPFv3允许每条链路上具有多个不同的实例，这是通过在OSPF包头里增加一个实例标识（Instance ID）区别不同的实例来实现的。一个分配了给定实例标识的实例将会丢弃那些与该实例标识不匹配的OSPF数据包。
- 取消了OSPF特有的认证 ——IPv6协议使用认证扩展报头，这是一个标准的认证过程。由于这个原因，OSPFv3不需要OSPFv3数据

包自己的认证，它只要使用IPv6的认证就可以了。

- 更灵活地处理未知**LSA**类型 ——在OSPFv2中总是丢弃未知的LSA类型，而OSPFv3可以把它们当作链路本地泛洪扩散范围，或者像它们被识别一样保存和泛洪扩散，但在它们自己的SPF算法中将被忽略。结果是，在OSPFv3中处理网络变化和继承新的特性比在OSPFv2中更容易。

9.1.2 OSPFv3的消息

OSPFv3和OSPFv2使用相同的协议号——89，当然OSPFv3作为一个IPv6的协议，更准确地说是使用设置为89的下一报头值。和OSPFv2一样，OSPFv3尽可能的使用多播。IPv6的AllSPFRouters多播地址为FF02::5，AllDRouters多播地址为FF02::6。它们都是链路本地的范围。读者在这里可以很容易地发现它们和OSPFv2的相应多播地址224.0.0.5和224.0.0.6的最后一些位是相似的。

OSPFv3同样使用相同的5种消息类型——Hello、DD、LS数据库请求、LS数据库更新和LS确认，这和OSPFv2中的一样，包括它们的编号也相同。如图9-1所示显示了OSPFv3的消息头部，这与OSPFv2消息头部稍微有些不同。当然，它的版本号不是2，而是3。但更重要的是，在它的报头里没有认证字段。正如前面所讨论的，OSPFv3使用IPv6数据包本身的扩展报头来进行认证，而不是利用它自己的认证进程。在这里，我们可以看到有一个实例标识，用来允许在同一个链路上运行多个OSPFv3实例。请注意，接口标识只在本地链路上具有意义，因为OSPFv3消息不能转发到始发它的链路之外。

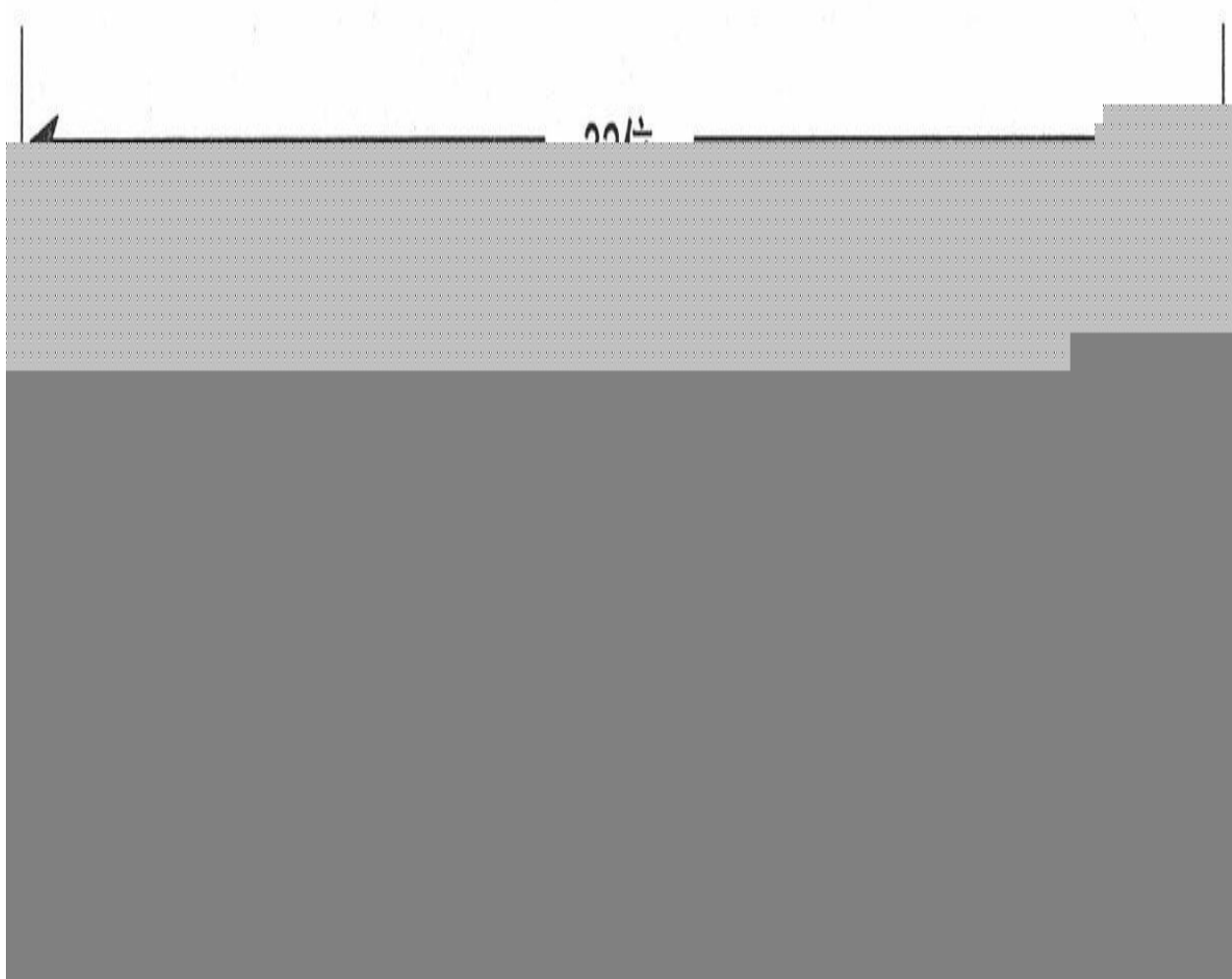


图9-10 SPfv3的包头格式

除了消息报头之外，只有两种OSPFv3消息的格式和OSPFv2对应的消息不同：Hello和数据库描述消息。

如图9-2所示，图中显示了OSPFv3的Hello消息格式。和OSPFv2不同，由于IPv6不需要网络掩码，这里的消息格式中没有网络掩码字段。除此以外，两个版本的数据包都具有相同的字段（报头以下）。可选项字段的大小增大到了24位，路由器无效时间间隔则从32位减少到了16位。这个字段意味着路由器无效时间间隔的理论最大值从43亿秒减少为65535秒。这个变化对于所运行的网络是比较小的，影响很小，甚至没有。在大多数普通的OSPF实现中所配置的最大路由器无效时间间隔总是65535秒，好的OSPF设计是不会接近于这个最大值的。因此，这个字段大小的变化仅仅是节约了无用的空间。

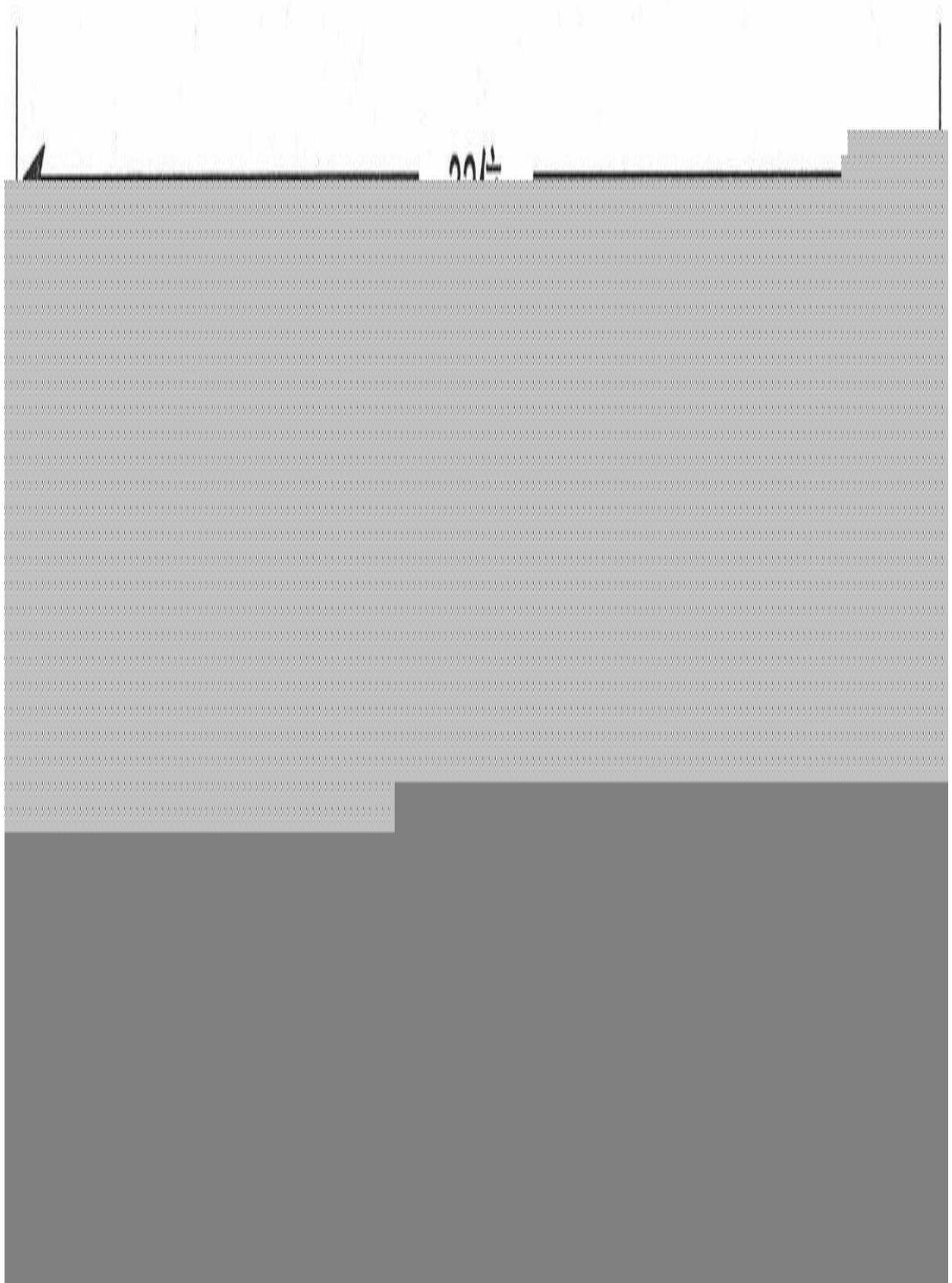


图9-2 OSPFv3的Hello消息格式

在图9-3中显示了OSPFv3数据库描述数据包的格式。与OSPFv2的对应部分所不同的仅仅是更大一点的可选项字段。

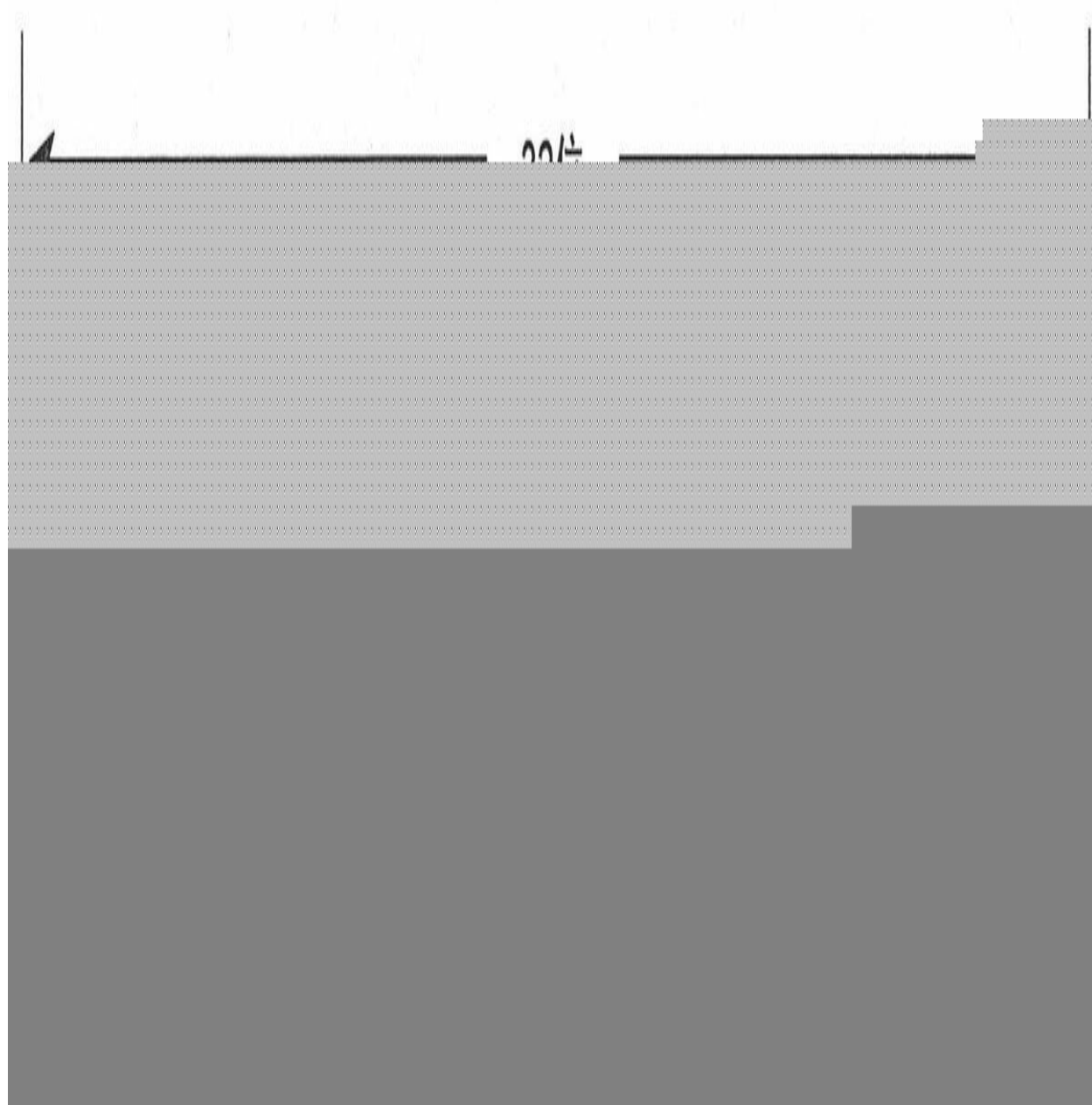


图9-3 OSPFv3数据库描述数据包的格式

其他3种类型的消息——链路状态请求、链路状态更新和链路状态确认

——的格式都和OSPFv2中对应的消息类型格式相同，因此在这一章将不再赘述。

9.1.3 OSPFv3的LSA概述

如图9-4所示，图中显示了OSPFv3的LSA报头。与图8-54中显示的OSPFv2的LSA报头对比，读者可以看出除了两处不同外，它们几乎是相同的。一是没有了可选项字段，二是链路状态类型字段的大小是16位，而不是OSPFv2中的8位类型字段。

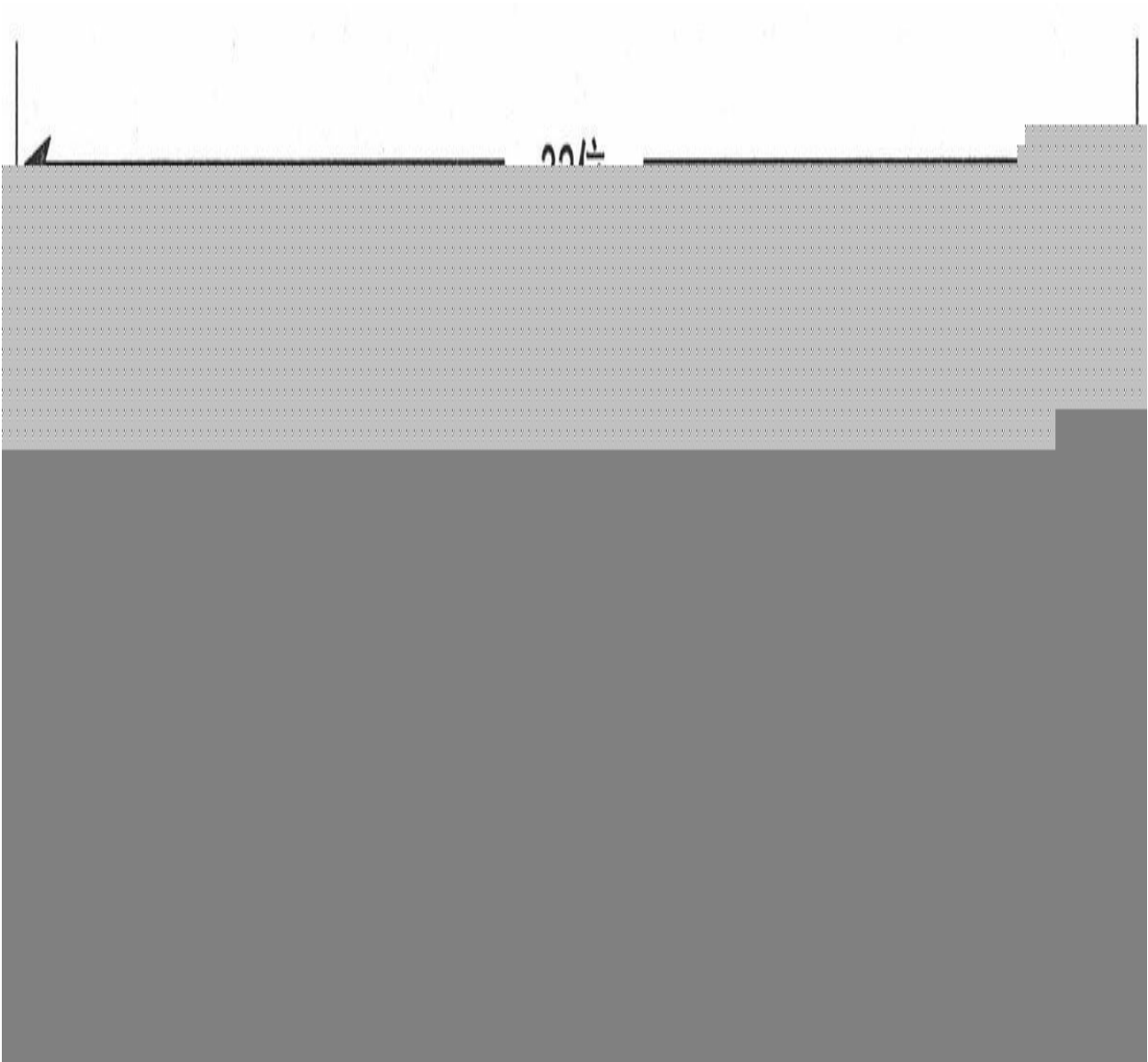


图9-4 OSPFv3的LSA报头格式

OSPFv3的LSA报头中链路状态类型字段加长的原因是因为它包含了3个前置位，如图9-5所示。

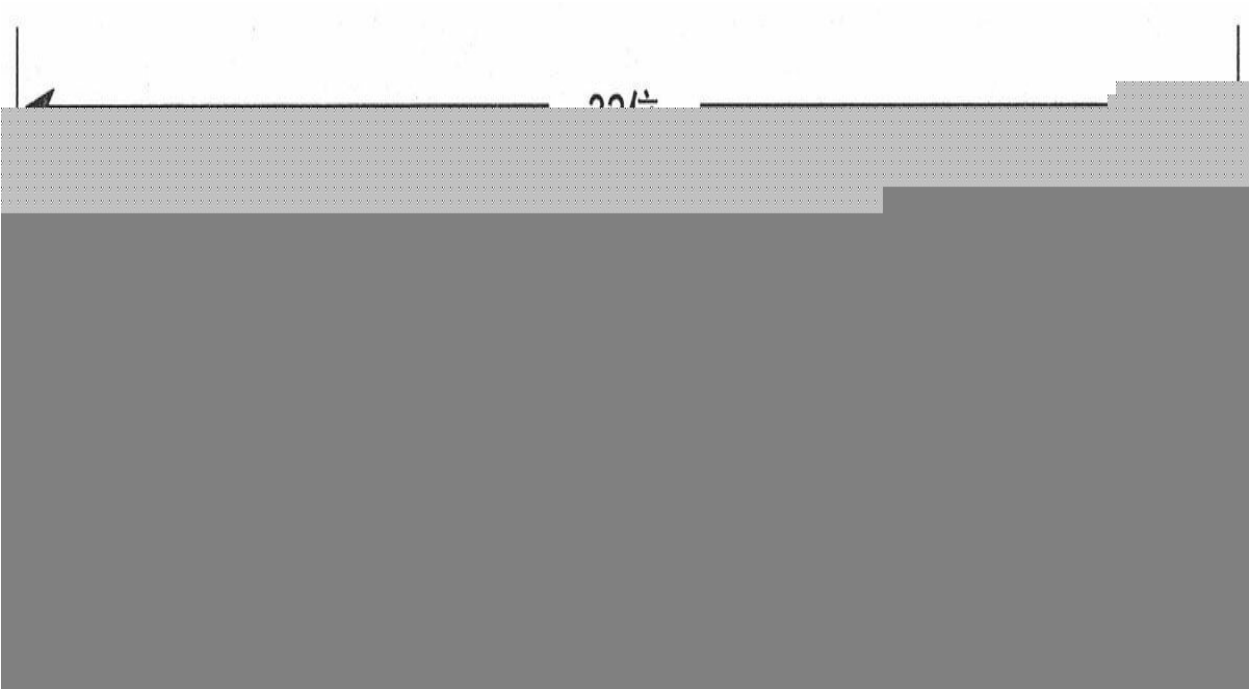


图9-5 OSPFv3的LSA报头的链路状态类型字段

U位 ——指出了一台路由器如果不能识别LSA的功能代码（LSA Function Code）时，应该如何处理该LSA。如果该位没有设置，未知的LSA将被作为链路本地泛洪扩散的范围处理。如果该位被设置，未知的LSA将会像被识别的LSA一样被保存和扩散。

S2和S1位 ——指出了LSA的泛洪扩散范围。在表9-1中列出了这两位可能的值和相对应的泛洪扩散范围。

表9-1 OSPFv3的LSA链路状态类型字段中的S位和它们对应的泛洪扩散范围

S2	S1	泛洪扩散范围
0	0	链路本地
0	1	区域
1	0	自主系统（AS）（路由选择域）
1	1	保留

LSA功能代码（LSA Function Code）是LS类型字段的最后13位，这和OSPFv2的类型 字段相一致。表9-2中显示了OSPFv3常用的LSA类型和它们对应的LS类型的值。如果读者翻译这些十六进制的值，将可以看到它们所有的U位缺省都是0。除了两种类型之外，其他所有的LSA的S位都标识为区域范围。两个例外是，AS外部LSA具有AS泛洪扩散范围，而链路LSA具有链路本地的泛洪扩散范围。大多数OSPFv3的LSA和OSPFv2相应的LSA具有相同的功能，这些OSPFv2的LSA和它们的类型也显示在表9-2中。

OSPFv3 LSA Type	OSPFv2 LSA Type
0x00000001	0x00000001
0x00000002	0x00000002
0x00000003	0x00000003
0x00000004	0x00000004
0x00000005	0x00000005
0x00000006	0x00000006
0x00000007	0x00000007
0x00000008	0x00000008
0x00000009	0x00000009
0x0000000A	0x0000000A
0x0000000B	0x0000000B
0x0000000C	0x0000000C
0x0000000D	0x0000000D
0x0000000E	0x0000000E
0x0000000F	0x0000000F
0x00000010	0x00000010
0x00000011	0x00000011
0x00000012	0x00000012
0x00000013	0x00000013
0x00000014	0x00000014
0x00000015	0x00000015
0x00000016	0x00000016
0x00000017	0x00000017
0x00000018	0x00000018
0x00000019	0x00000019
0x0000001A	0x0000001A
0x0000001B	0x0000001B
0x0000001C	0x0000001C
0x0000001D	0x0000001D
0x0000001E	0x0000001E
0x0000001F	0x0000001F
0x00000020	0x00000020
0x00000021	0x00000021
0x00000022	0x00000022
0x00000023	0x00000023
0x00000024	0x00000024
0x00000025	0x00000025
0x00000026	0x00000026
0x00000027	0x00000027
0x00000028	0x00000028
0x00000029	0x00000029
0x0000002A	0x0000002A
0x0000002B	0x0000002B
0x0000002C	0x0000002C
0x0000002D	0x0000002D
0x0000002E	0x0000002E
0x0000002F	0x0000002F
0x00000030	0x00000030
0x00000031	0x00000031
0x00000032	0x00000032
0x00000033	0x00000033
0x00000034	0x00000034
0x00000035	0x00000035
0x00000036	0x00000036
0x00000037	0x00000037
0x00000038	0x00000038
0x00000039	0x00000039
0x0000003A	0x0000003A
0x0000003B	0x0000003B
0x0000003C	0x0000003C
0x0000003D	0x0000003D
0x0000003E	0x0000003E
0x0000003F	0x0000003F
0x00000040	0x00000040
0x00000041	0x00000041
0x00000042	0x00000042
0x00000043	0x00000043
0x00000044	0x00000044
0x00000045	0x00000045
0x00000046	0x00000046
0x00000047	0x00000047
0x00000048	0x00000048
0x00000049	0x00000049
0x0000004A	0x0000004A
0x0000004B	0x0000004B
0x0000004C	0x0000004C
0x0000004D	0x0000004D
0x0000004E	0x0000004E
0x0000004F	0x0000004F
0x00000050	0x00000050
0x00000051	0x00000051
0x00000052	0x00000052
0x00000053	0x00000053
0x00000054	0x00000054
0x00000055	0x00000055
0x00000056	0x00000056
0x00000057	0x00000057
0x00000058	0x00000058
0x00000059	0x00000059
0x0000005A	0x0000005A
0x0000005B	0x0000005B
0x0000005C	0x0000005C
0x0000005D	0x0000005D
0x0000005E	0x0000005E
0x0000005F	0x0000005F
0x00000060	0x00000060
0x00000061	0x00000061
0x00000062	0x00000062
0x00000063	0x00000063
0x00000064	0x00000064
0x00000065	0x00000065
0x00000066	0x00000066
0x00000067	0x00000067
0x00000068	0x00000068
0x00000069	0x00000069
0x0000006A	0x0000006A
0x0000006B	0x0000006B
0x0000006C	0x0000006C
0x0000006D	0x0000006D
0x0000006E	0x0000006E
0x0000006F	0x0000006F
0x00000070	0x00000070
0x00000071	0x00000071
0x00000072	0x00000072
0x00000073	0x00000073
0x00000074	0x00000074
0x00000075	0x00000075
0x00000076	0x00000076
0x00000077	0x00000077
0x00000078	0x00000078
0x00000079	0x00000079
0x0000007A	0x0000007A
0x0000007B	0x0000007B
0x0000007C	0x0000007C
0x0000007D	0x0000007D
0x0000007E	0x0000007E
0x0000007F	0x0000007F
0x00000080	0x00000080
0x00000081	0x00000081
0x00000082	0x00000082
0x00000083	0x00000083
0x00000084	0x00000084
0x00000085	0x00000085
0x00000086	0x00000086
0x00000087	0x00000087
0x00000088	0x00000088
0x00000089	0x00000089
0x0000008A	0x0000008A
0x0000008B	0x0000008B
0x0000008C	0x0000008C
0x0000008D	0x0000008D
0x0000008E	0x0000008E
0x0000008F	0x0000008F
0x00000090	0x00000090
0x00000091	0x00000091
0x00000092	0x00000092
0x00000093	0x00000093
0x00000094	0x00000094
0x00000095	0x00000095
0x00000096	0x00000096
0x00000097	0x00000097
0x00000098	0x00000098
0x00000099	0x00000099
0x0000009A	0x0000009A
0x0000009B	0x0000009B
0x0000009C	0x0000009C
0x0000009D	0x0000009D
0x0000009E	0x0000009E
0x0000009F	0x0000009F
0x000000A0	0x000000A0
0x000000A1	0x000000A1
0x000000A2	0x000000A2
0x000000A3	0x000000A3
0x000000A4	0x000000A4
0x000000A5	0x000000A5
0x000000A6	0x000000A6
0x000000A7	0x000000A7
0x000000A8	0x000000A8
0x000000A9	0x000000A9
0x000000AA	0x000000AA
0x000000AB	0x000000AB
0x000000AC	0x000000AC
0x000000AD	0x000000AD
0x000000AE	0x000000AE
0x000000AF	0x000000AF
0x000000B0	0x000000B0
0x000000B1	0x000000B1
0x000000B2	0x000000B2
0x000000B3	0x000000B3
0x000000B4	0x000000B4
0x000000B5	0x000000B5
0x000000B6	0x000000B6
0x000000B7	0x000000B7
0x000000B8	0x000000B8
0x000000B9	0x000000B9
0x000000BA	0x000000BA
0x000000BB	0x000000BB
0x000000BC	0x000000BC
0x000000BD	0x000000BD
0x000000BE	0x000000BE
0x000000BF	0x000000BF
0x000000C0	0x000000C0
0x000000C1	0x000000C1
0x000000C2	0x000000C2
0x000000C3	0x000000C3
0x000000C4	0x000000C4
0x000000C5	0x000000C5
0x000000C6	0x000000C6
0x000000C7	0x000000C7
0x000000C8	0x000000C8
0x000000C9	0x000000C9
0x000000CA	0x000000CA
0x000000CB	0x000000CB
0x000000CC	0x000000CC
0x000000CD	0x000000CD
0x000000CE	0x000000CE
0x000000CF	0x000000CF
0x000000D0	0x000000D0
0x000000D1	0x000000D1
0x000000D2	0x000000D2
0x000000D3	0x000000D3
0x000000D4	0x000000D4
0x000000D5	0x000000D5
0x000000D6	0x000000D6
0x000000D7	0x000000D7
0x000000D8	0x000000D8
0x000000D9	0x000000D9
0x000000DA	0x000000DA
0x000000DB	0x000000DB
0x000000DC	0x000000DC
0x000000DD	0x000000DD
0x000000DE	0x000000DE
0x000000DF	0x000000DF
0x000000E0	0x000000E0
0x000000E1	0x000000E1
0x000000E2	0x000000E2
0x000000E3	0x000000E3
0x000000E4	0x000000E4
0x000000E5	0x000000E5
0x000000E6	0x000000E6
0x000000E7	0x000000E7
0x000000E8	0x000000E8
0x000000E9	0x000000E9
0x000000EA	0x000000EA
0x000000EB	0x000000EB
0x000000EC	0x000000EC
0x000000ED	0x000000ED
0x000000EE	0x000000EE
0x000000EF	0x000000EF
0x000000F0	0x000000F0
0x000000F1	0x000000F1
0x000000F2	0x000000F2
0x000000F3	0x000000F3
0x000000F4	0x000000F4
0x000000F5	0x000000F5
0x000000F6	0x000000F6
0x000000F7	0x000000F7
0x000000F8	0x000000F8
0x000000F9	0x000000F9
0x000000FA	0x000000FA
0x000000FB	0x000000FB
0x000000FC	0x000000FC
0x000000FD	0x000000FD
0x000000FE	0x000000FE
0x000000FF	0x000000FF

表9-2 OSPFv3的LSA类型和它们在OSPFv2中对应部分

虽然OSPFv3和OSPFv2的路由器LSA与网络LSA具有相同的名字，但它们的两个版本里有很大的区别。特别地，OSPFv3的路由器与网络LSA并不通告前缀。在协议的扩展性方面，这是一个重要的改进。正

如读者在第8章所了解的，这些LSA把路由器首先看作SPF树上的一个节点。因此，当路由器LSA或网络LSA扩散时，就假定网络拓扑已经发生了变化，区域内的所有路由器在收到LSA后就要重新运行SPF。但是，由于OSPFv2路由器也使用这些LSA通告它们所连接的子网，如果子网发生变化，相应的LSA也必须进行扩散以便通告这种变化。即使是像一些并不影响SPF拓扑的地址变化，收到路由器LSA或网络LSA也总会触发一个SPF算法的运行。这在那些具有很多定期变化的末梢链路的边界或接入路由器上特别容易出现问題。

OSPFv3在路由器LSA和网络LSA中取消了通告前缀的功能，而是把这项功能放入新的区域内前缀LSA当中。这样路由器LSA和网络LSA对于SPF来说就只代表路由器的节点信息，并且只有当与SPF算法相关的信息发生变化时它们才会进行扩散。如果前缀发生变化，或者末梢链路的状态发生改变，这些信息将在区域内前缀LSA中进行扩散，而区域内前缀LSA不会触发SPF的运行。

路由器与网络LSA在两个OSPF版本之间的另外一个不同之处是，一些只与直连邻居相关的信息交换。OSPFv2把这个信息放入路由器与网络LSA里，虽然只有直连邻居关心这个信息，但它却伴随着路由器与网络LSA在整个区域里进行扩散。OSPFv3则把这个邻居特有的信息放入新的链路LSA中，链路LSA只具有链路本地的扩散范围。链路LSA的引入确实提高了OSPFv2的效率，尽管这种提高并不明显。

区域间前缀、区域间路由器和类型7 LSA的名字虽然变化了，但它们分别与OSPFv2中对应的网络汇总、ASBR汇总和NSSA LSA具有相同的功能。AS外部和组成员LSA的名字和功能在两个OSPF版本中都是一样的。

9.1.4 OSPFv3的LSA格式

这一小节将详细讲述表9-2中前面5种OSPFv3的LSA类型和两种新的LSA类型的格式。虽然有一个在多播OSPF（MOSPF）中专用的组成员LSA，但多播协议并不在本书的讲述范围内，因此也不详细讨论这个LSA。在大多数情况下，这里仅仅讨论与OSPFv2的对应部分有所不同的LSA字段特性。建议读者将具有对应的OSPFv2 LSA的LSA与第8章中的内容比较。

1. 路由器LSA

如图9-6所示，图中显示了OSPFv3路由器LSA的格式。正如前面章节所讨论的，在路由器LSA中没有包含前缀信息，而在OSPFv2版本对应的路由器LSA中是包含的。OSPFv3的路由器LSA仅仅描述始发路由器和与连接到邻居的链路，用于SPF的计算。前缀信息则在区域内前缀LSA中携带。

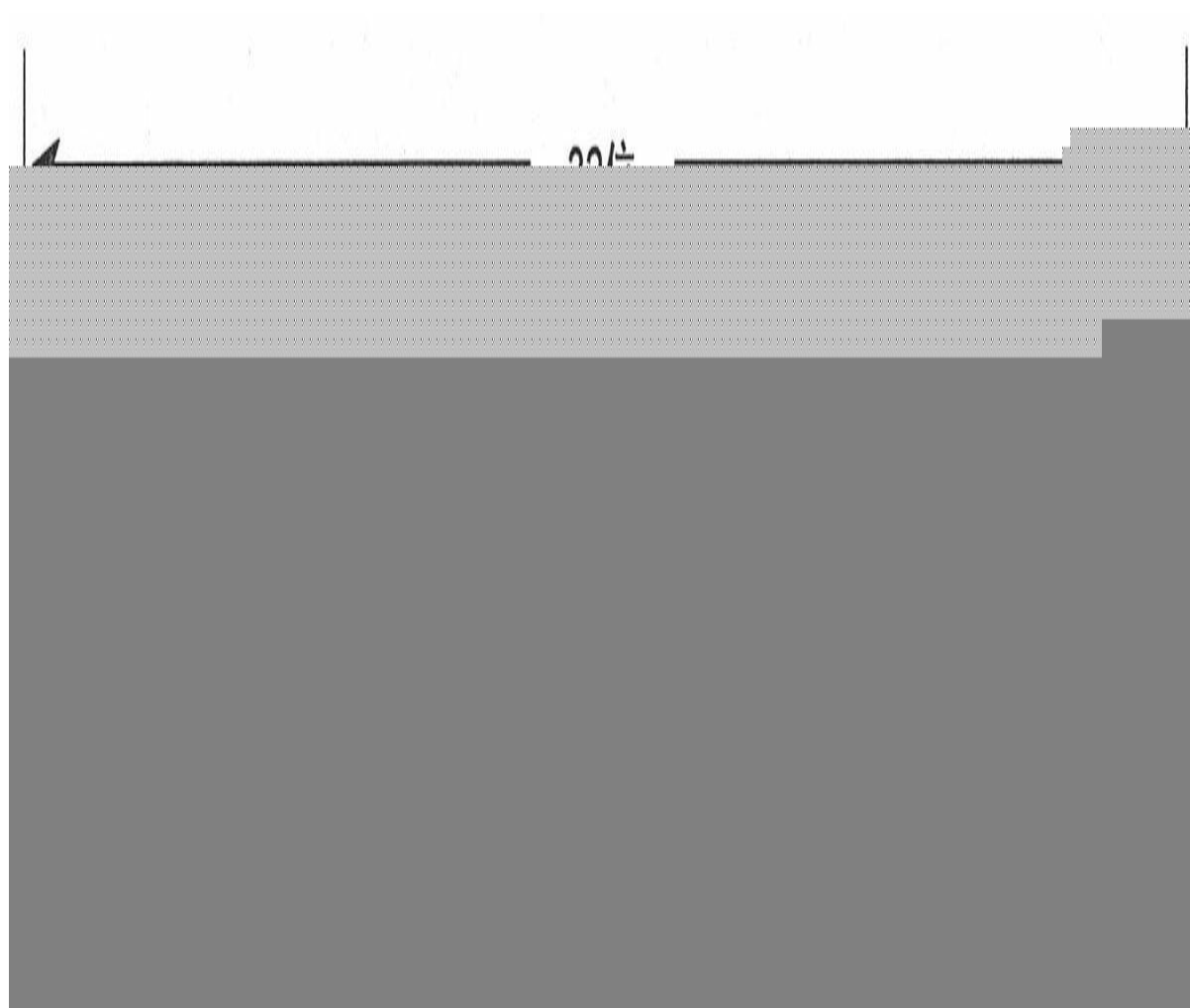


图9-6 OSPFv3的路由器LSA格式

这里也要注意，在OSPFv2中已经淘汰的ToS度量字段也没有包含在这个LSA中。

- 可选项字段 ——OSPFv3与OSPFv2的可选项字段相比，它在LSA格式中的位置不同，长度也加长了（24位），但它们承担的作用是相同

的，都是标识一些可选的性能。在后面的章节中将会分别讲述各种数据包和LSA中携带的可选项字段。

跟在可选项字段之后的是一组多次出现用来描述每一种相关联的接口的字段集合。

- 类型（**Type**）——是指接口的类型。表9-3中列出了可能的接口类型的值。

表9-3 路由器LSA的类型字段中指定的接口类型

类型	描述
1	点到点连接到另一台路由器
2	连接到一个传送网络（transit network）
3	保留
4	虚链路

- 度量（**Metric**）——是指接口的出站代价（outbound cost）。
- 接口 **ID** ——是一个32位的值，用来把该接口与始发路由器上其他的接口区分开来。
- 邻居接口**ID**（**Neighbor Interface ID**）——是在Hello数据包中链路上邻居通告的接口 ID，或者是类型2的链路上链路的指定路由器（DR）的接口 ID。
- 邻居路由器**ID**（**Neighbor Router ID**）——是指邻居的RID，或者是类型2的链路上指定路由器（DR）的RID。

2. 网络LSA

如图9-7所示，图中显示了OSPFv3的网络LSA格式。就其功能而言，它与OSPFv2中的网络LSA是相同的（始发于DR，代表一个伪节点，假定到与它直接相连的邻居的代价为0，等等）。它们之间惟一的重要不同之处是可选项字段的位置不同，OSPFv3中取消了在IPv6协议中没有意义的网络掩码字段。

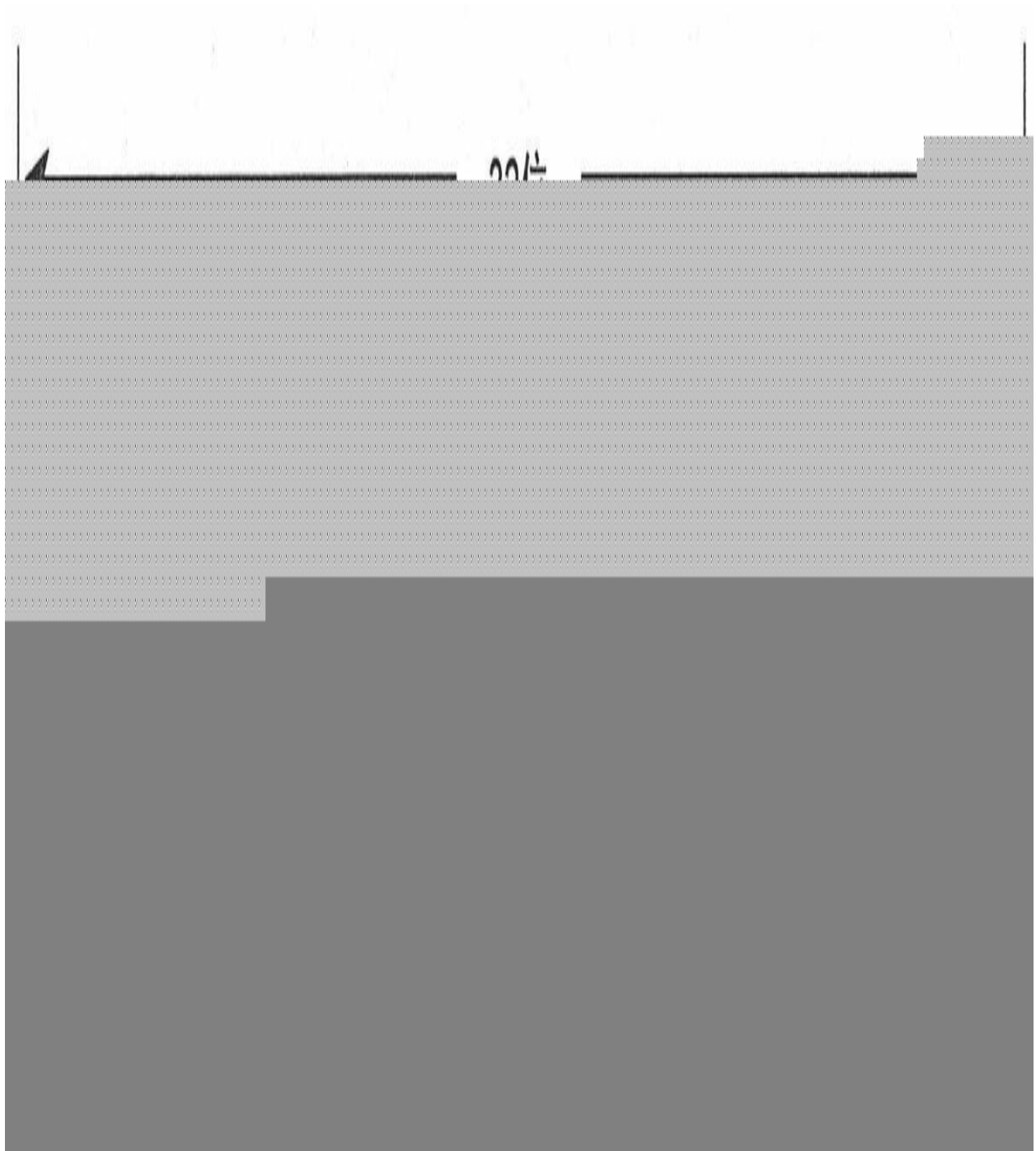


图9-7 OSPFv3的网络LSA格式

3. 区域间前缀LSA

如图9-8所示，图中显示了OSPFv3区域间前缀LSA的格式。它的作用和OSPFv2中类型3汇总LSA相同——由ABR路由器始发这种LSA到一个区

域，通告该区域外部但仍然属于它的OSPF域内的网络。在OSPFv3中只是名字变得更具有描述性。ABR路由器必须为每一个在区域内被通告的IPv6前缀发起一个单独的区域间前缀LSA。ABR路由器也能够始发一个区域间前缀LSA向一个末梢区域通告一条缺省路由。

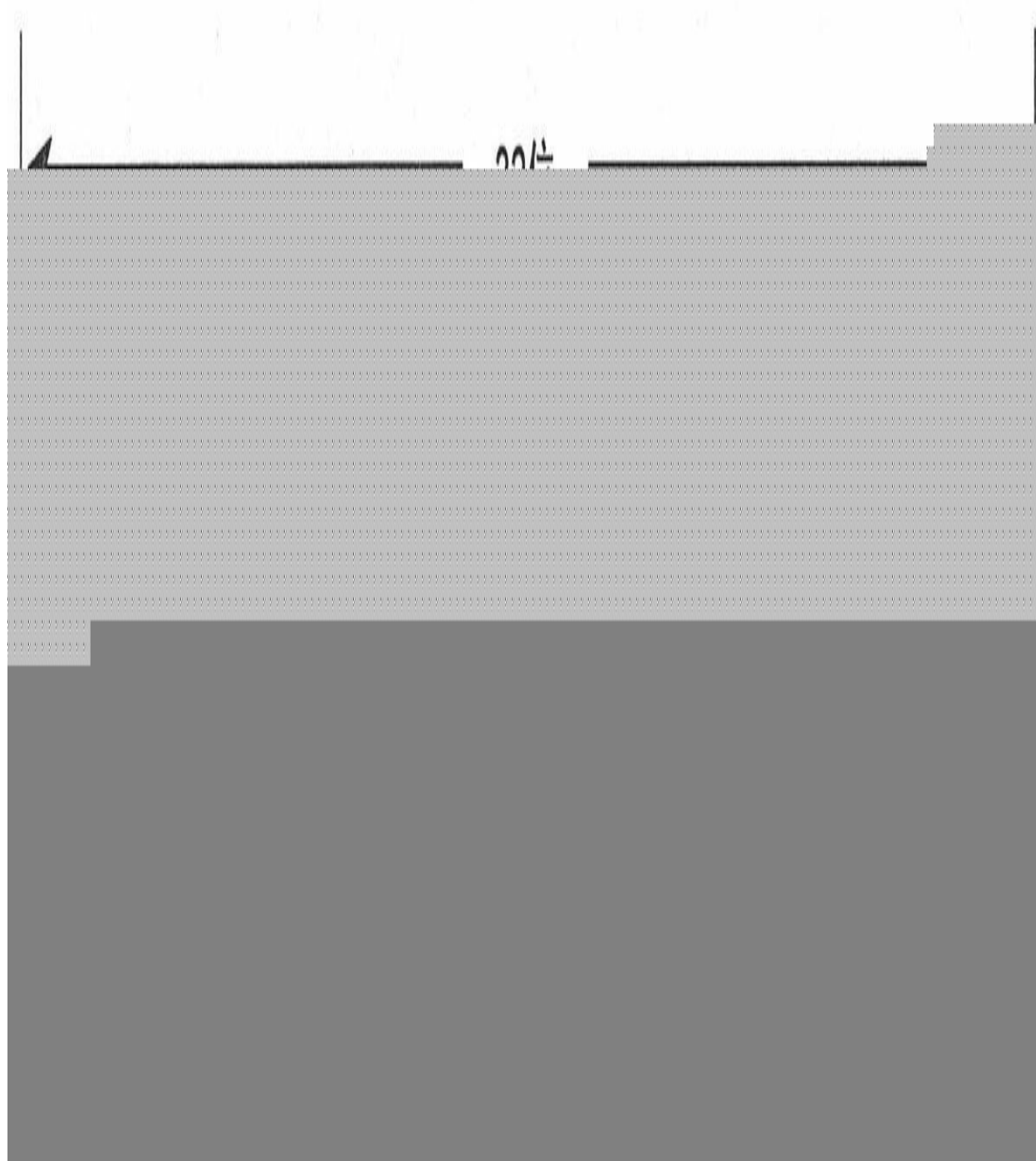


图9-8 OSPFv3区域间前缀LSA的格式

4. 区域间路由器LSA

在OSPFv3中区域间路由器LSA的功能和OSPFv2中类型4汇总LSA所承担的功能是相同的。ABR向一个区域内始发一条区域间路由器LSA，用来通告一个在该区域外的ASBR路由器。对于所通告的每一个ASBR，ABR都需要始发单独的区域间路由器LSA。

如图9-9所示，图中显示了OSPFv3区域间路由器LSA的格式。即使OSPFv2类型3和类型4汇总LSA具有相同的格式，读者通过比较图9-8和图9-9能够看到区域间网络LSA和区域间路由器LSA是不同的。字段是自描述的。

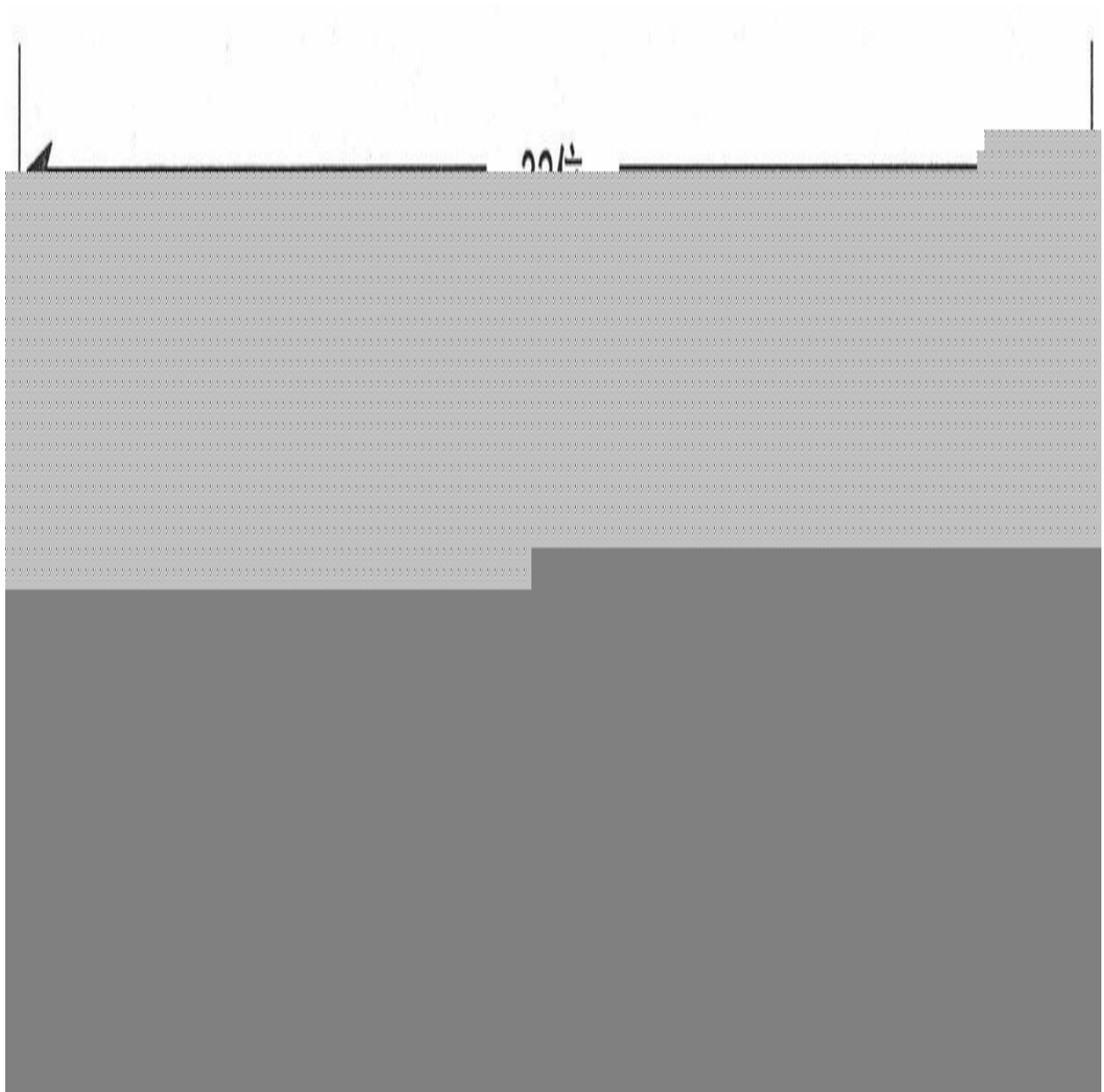


图9-9 OSPFv3区域间路由器LSA的格式

- 可选项 ——表示ASBR的可选性能。
- 度量 ——表示ASBR的代价。
- 目的路由器 ID (**Destination Router ID**) ——是指ASBR的RID。

5. AS外部LSA

和OSPFv2中一样，AS外部LSA通告处于OSPF路由选择域外部的前缀。对于每一条 需要通告的外部前缀都需要一条这样的LSA。但是，OSPFv3中AS外部LSA的格式（参见图9-10）不同于OSPFv2中的对应部分。

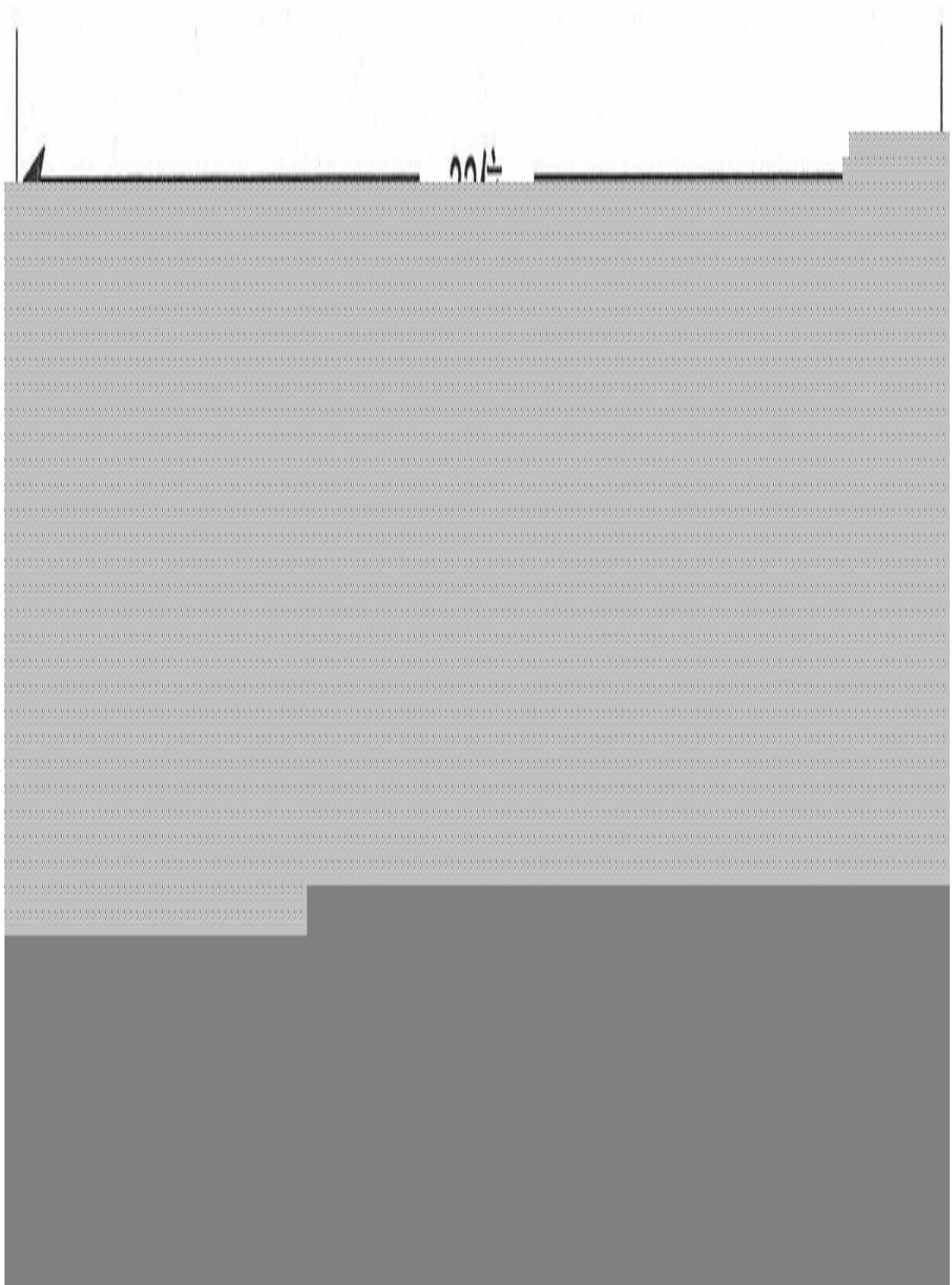


图9-10 OSPFv3 AS外部LSA的格式

- **E标记** ——在OSPFv3版的LSA中它的功能与它在OSPFv2版的LSA中的功能相同。如果设置该位，表示度量是类型2的外部度量。如果没有设置该位，表示度量是类型1的外部度量。
- **F标记** ——当设置该位时，表示该LSA中包含一个转发地址。
- **T标记** ——当设置该位时，表示该LSA中包含一个外部路由标记。
- **度量** ——顾名思义，表示路由的代价。无论它是类型1还是类型2，都依赖于E标记的值。
- **前缀长度、前缀可选项和地址前缀** ——完整地描述了嵌套的前缀。
- **转发地址** ——如果包含这一项，它是一个完整的128位IPv6地址，用来表示目的前缀的下一跳地址。只有设置了F标记才会包含这一项。
- **外部路由标记（External Route Tag）** ——如果包含这一项，它所承担的作用和OSPFv2中AS外部LSA的外部路由标记字段的作用是相同的。只有设置了T标记才会包含这一项。
- **引用链路状态ID（Referenced Link State ID）和引用链路状态类型（Referenced Link State Type）** ——如果包含这两项，将允许该前缀的附加信息包含在另一个LSA中。如果使用了这两个字段，它们将描述携带附加信息的LSA的链路状态ID和类型。所引用的LSA的通告路由器字段也必须与该AS外部LSA中通告路由器字段的值相匹配。如果带有外部路由标记，这些附加信息就和OSPFv3本身无关，它们可以在边界路由器之间穿越整个OSPF域。如果这个功能项没有使用，那么引用链路状态类型字段就设置为全0。

6. 链路LSA

链路LSA只有用于两个直接相连的邻居之间的信息通信才是有意义的。

如图9-11所示，图中显示了链路LSA的格式。每一个与路由器相连并属于同一个OSPFv3域的链路都要始发一个单独的链路LSA，而且由于它属于链路本地扩散的范围，收到它的路由器从来不会把它转发到其他链路上去。

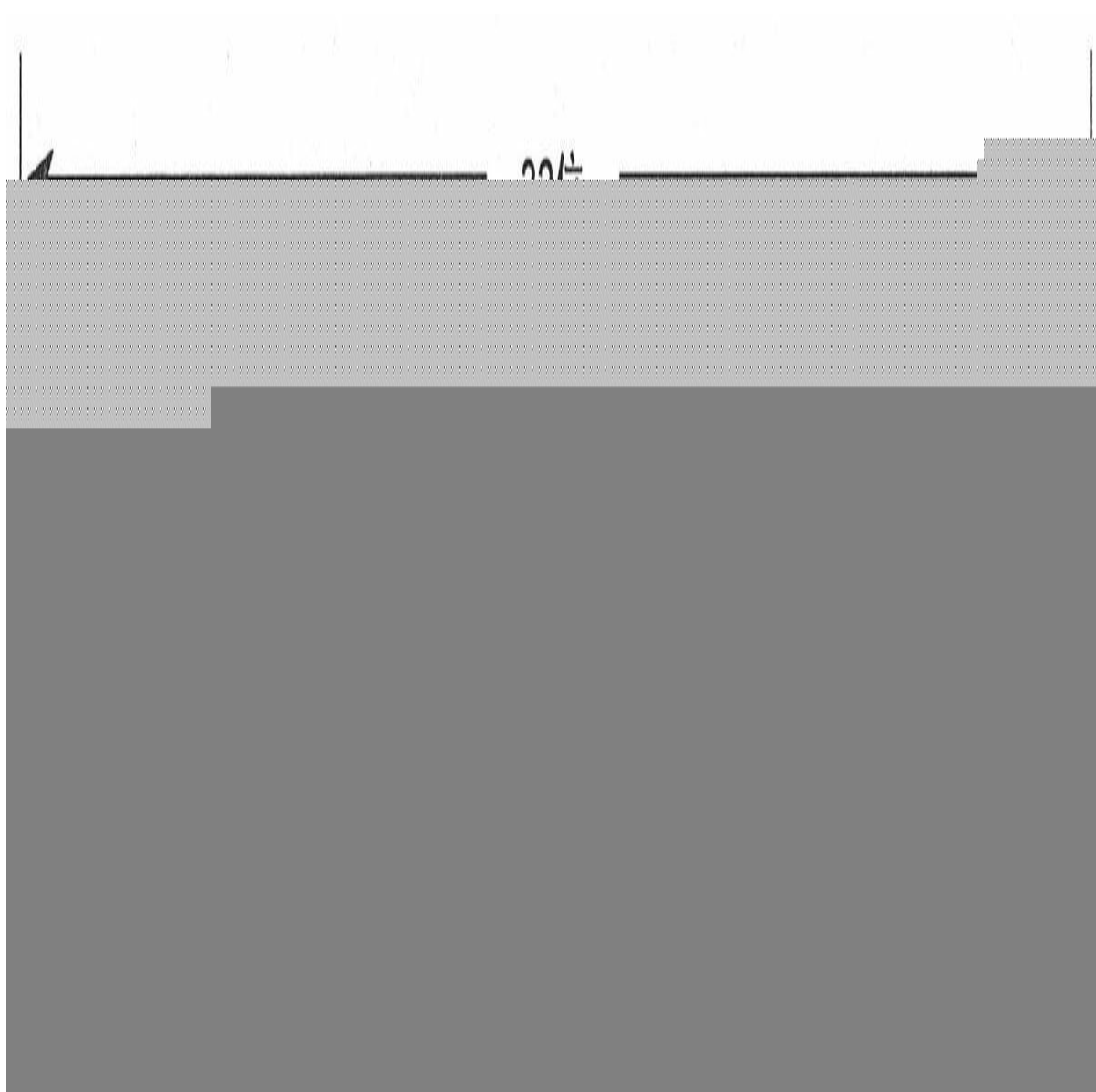


图9-11 OSPFv3链路LSA的格式

链路LSA具有以下3个功能：

（1）链路LSA向与这条链路相连的其他所有路由器提供始发路由器的链路本地地址。

(2) 链路LSA提供了与这条链路有关的IPv6前缀列表。

(3) 链路LSA提供了这条链路始发的网络LSA有关的一组可选位的集合。

- 路由器优先级 (**Router Priority**) ——标识了分配给始发路由器接口的路由器优先级。
- 可选项 ——表示始发路由器应该在为该链路发起的网络LSA中设置的可选项位。这同样是一个24位的可选项字段，由OSPFv3 Hello和DD数据包，以及许多OSPFv3 LSA携带。可选项字段将在后面的章节中详细讲述。
- 链路本地前缀地址 ——用来标识始发路由器与该链路相连的接口的128位链路本地前缀。
- 前缀数目 ——是指在LSA中包含的IPv6前缀的数量，这些前缀通过下面讲述的前缀长度、前缀可选项和地址前缀字段来描述。
- 前缀长度、前缀可选项和地址前缀 ——用来描述一台或多台始发路由器上与链路相连的IPv6前缀。这个字段集合不仅用于链路LSA，还用于区域内前缀、区域间前缀和AS外部LSA。所通告的前缀的长度可以是0~128之间的任意值。当前缀不是一个32位的偶数倍时，它就会被填充0来达到地址前缀字段的32位边界。前缀长度字段指定了未填充的前缀的长度，以位数为单位。前缀可选项字段如图9-12所示，它指定了在路由选择计算期间可选的操作。

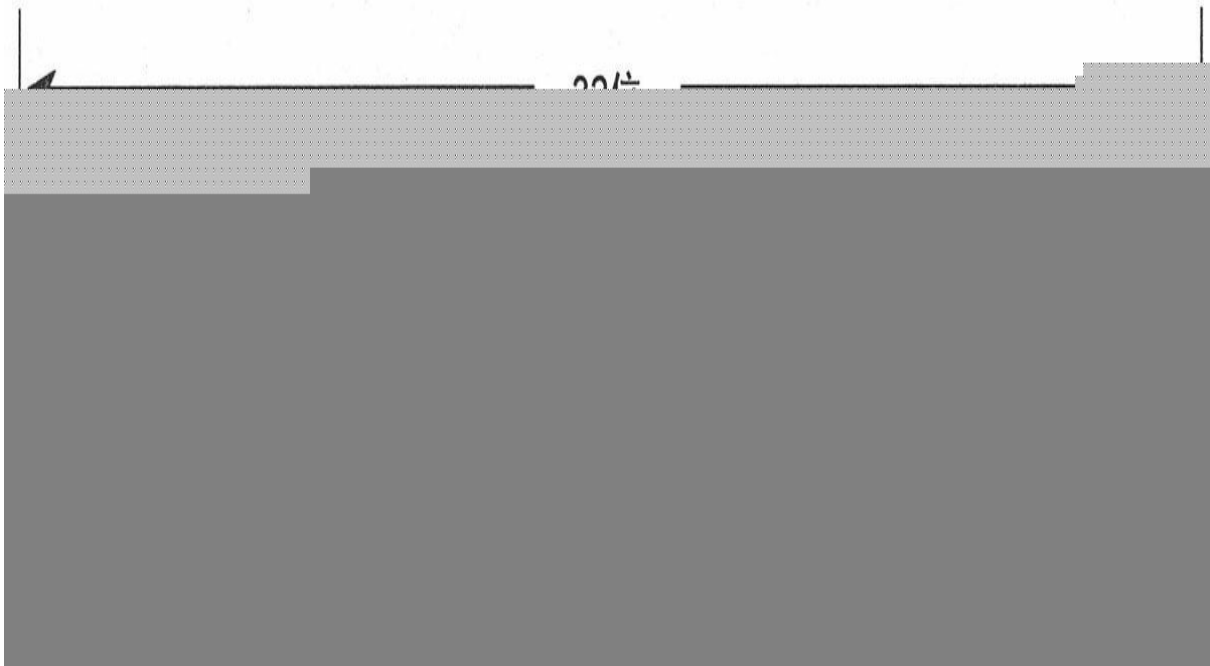


图9-12 前缀可选项字段

- 传播位（**Propagate, P**）——传播位设置在NSSA区域边界重新通告的NSSA区域前缀上。
- 多播位（**MC**）——如果设置该位，表示应该在多播路由选择计算中包含该前缀。
- 本地地址位（**Local Address, LA**）——如果设置该位，表示该前缀是一个通告路由器的接口地址。
- 无单播位（**No Unicast, NU**）——如果设置该位，表示该前缀应该从单播路由选择计算中排除。

7. 区域内前缀LSA

如图9-13所示，图中显示了区域内前缀LSA的格式。回忆前面章节有关这个LSA的讨论，在OSPFv2版本中由路由器与网络LSA携带的前缀信息在OSPFv3版本中则由区域内前缀LSA携带。前缀的变化——包括增加和删除——都不会以任何方式影响SPF树的分支变化。但是OSPFv2在路由器与网络LSA中通告前缀，无论前缀是否发生变化，都会引起区域内所有路由器进行SPF的计算。在OSPFv3中，当一条链路或它的前缀发生

变化时，相连的路由器只会始发一台区域内前缀LSA在整个区域内扩散这个信息。这个LSA并不会触发SPF计算，收到该LSA的路由器只是简单地把这个新的前缀信息与始发路由器关联起来。在OSPFv3中，路由器与网络LSA只用来提供拓扑的信息服务。结果是，这个新的LSA将使OSPFv3在处理存在大量频繁变化前缀的网络方面具有更好的扩展性。

- 前缀数目 ——表示该LSA中包含的前缀的数量。
- 引用链路状态ID（**Referenced Link State ID**）、引用链路状态类型（**Referenced Link State Type**）和引用通告路由器（**Reference Advertising Router**）——用来标识与所包含的前缀相关联的路由器或网络LSA。

如果一个前缀与一个路由器LSA关联，那么引用链路状态类型为1，引用链路状态ID为0，而引用通告路由器的值则为始发路由器的RID。如果一个前缀应该与一个网络LSA关联，那么引用链路状态类型为2，引用链路状态ID为该链路的DR的接口ID^[1]，而引用通告路由器的值则为DR路由器的RID。

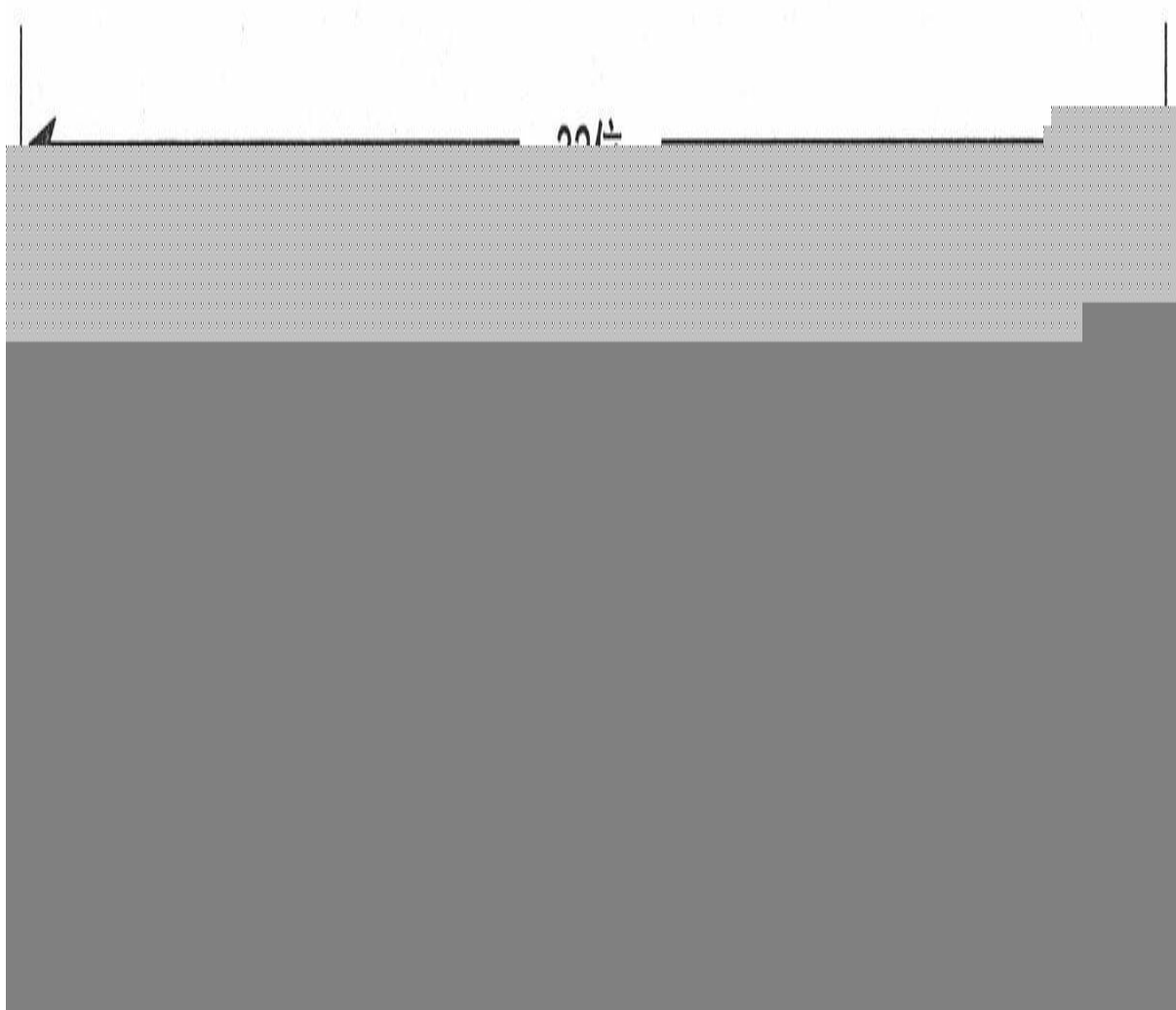


图9-13 区域内前缀LSA的格式

正如前面讲述的链路LSA的前缀一样，这里的每一个前缀也都通过一组前缀长度、前缀可选项和地址前缀字段来描述。除了这3个字段还增加了一个度量字段，用来表示前缀的代价。

8. 可选项字段

可选项字段的长度为24位，它用来指定始发路由器的可选性能；在路由器、网络、区域间路由器，以及链路LSA中都可以携带这个字段。这个字段还可以在Hello与数据库描述数据包中携带。如图9-14所示，图中显示了可选项字段的格式。在撰写本书的时候，只有最右边的6位已经定义作为可选标记，并且大多数的标记是读者在学习OSPFv2时已经熟悉

的。OSPFv3路由器忽略未被识别的可选项标记。

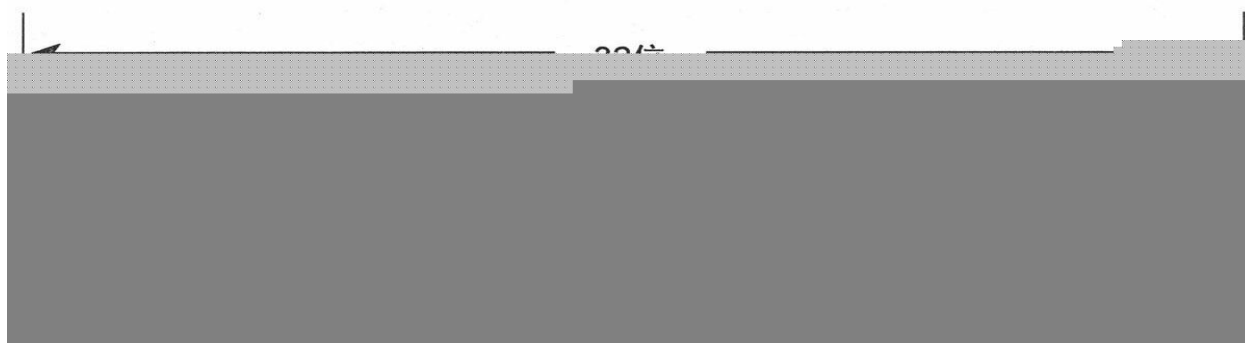


图9-14 可选项字段的格式

- **DC位** ——表示具有支持按需电路的能力。
- **R位** ——指明始发路由器是否是一台有效的路由器。当该位清0时，传送该始发节点的路由将不被计算。所以R标记增加了一个类似于IS-IS协议中超载位（OL位）的功能，超载位将在第10章中讲述。
- **N位** ——表示支持NSSA LSA。
- **MC位** ——表示支持MOSPF协议。
- **E位** ——表示对于末梢区域的形成，AS外部LSA是怎样进行扩散的。
- **V6位** ——如果该位清0，表示应该从IPv6路由选择计算中排除该路由器或链路。

9.2 OSPFv3的配置

OSPFv3的配置选项和OSPFv2中的相关选项基本相同。在OSPFv3中也需要指定进程ID和区域。在进程中需要包括接口和地址。区域可以被定义为末梢、完全末梢或者NSSA区域。在区域之间可以对前缀进行汇总。OSPFv3能够在按需电路和NBMA网络中配置。OSPFv3的大多数配置都和OSPFv2中的配置一样；在某些情况下需要增加IPv6关键字，而且IPv6的前缀或地址代替IPv4中的子网或地址。本节讲述的案例研究将阐述OSPFv3与OSPFv2不同的配置，以及一些配置非常相似但需要着重强调的地方。

9.2.1 案例研究：OSPFv3的基本配置

除了两点例外，OSPFv3的配置和OSPFv2的配置基本是相同的。回忆一下第8章中在路由器和接口上进行OSPFv2配置的步骤。首先，要使用**router ospf** 命令创建一个OSPF的进程。接着，要使用**network area** 命令指定一个覆盖需要运行OSPF协议的接口地址的地址范围。如果一个接口配置了属于由**network area** 命令指定的地址范围内的IPv4地址，那么这个接口将会运行OSPF协议。如果一个接口配置的IPv4地址不属于指定的地址范围，那么这个接口或这个地址（接口上配置有多个IPv4地址的情况）都将不会参与到OSPFv2进程的运行当中。支持IPv6的OSPFv3协议是通过在路由器的接口配置模式下指定一个OSPF进程ID和一个区域激活的。如果一个OSPFv3进程还没有创建，那么它会自动地创建。配置在路由器接口上的所有IPv6地址都被包含在这个指定的OSPF进程中运行。如图9-15所示，图中显示了一个运行OSPFv3协议的网络。

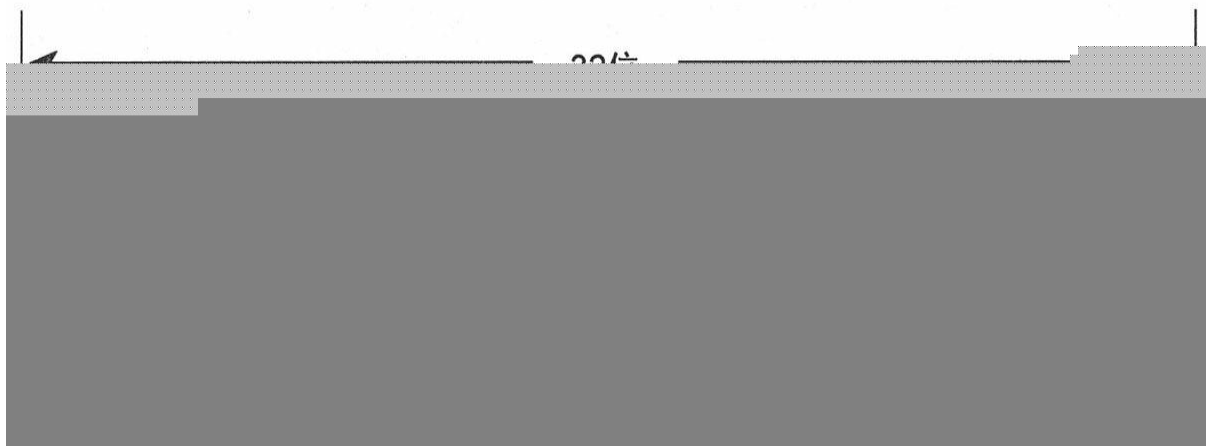
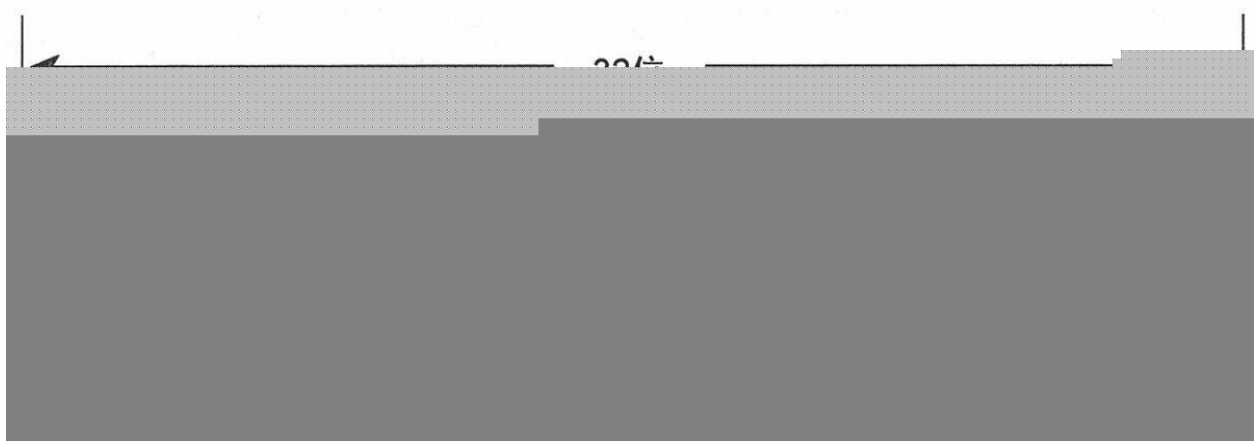


图9-15 一个OSPFv3网络的实例

路由器Hedwig与Pigwidgeon的配置参见示例9-1与示例9-2。

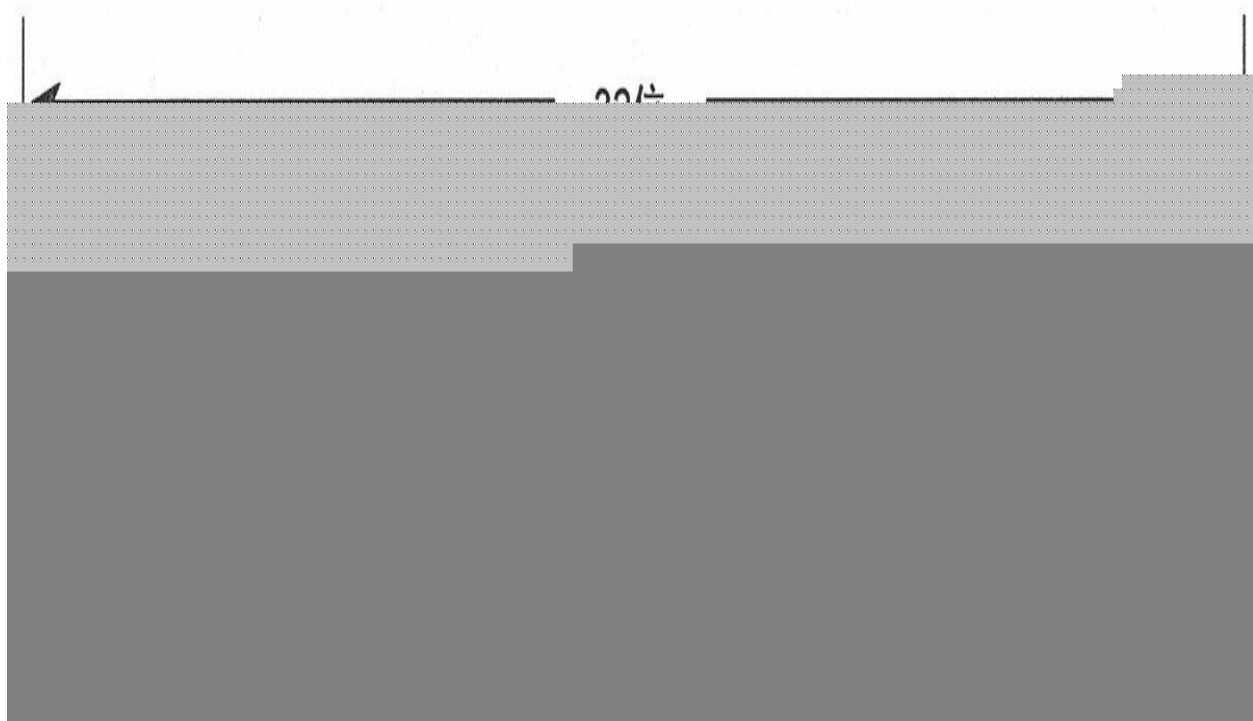
示例9-1 使用接口配置模式的命令**ipv6 ospf 1 area 1**在路由器**Hedwig**上配置**OSPFv3**



示例9-2 路由器**Pigwidgeon**上的**OSPFv3**配置

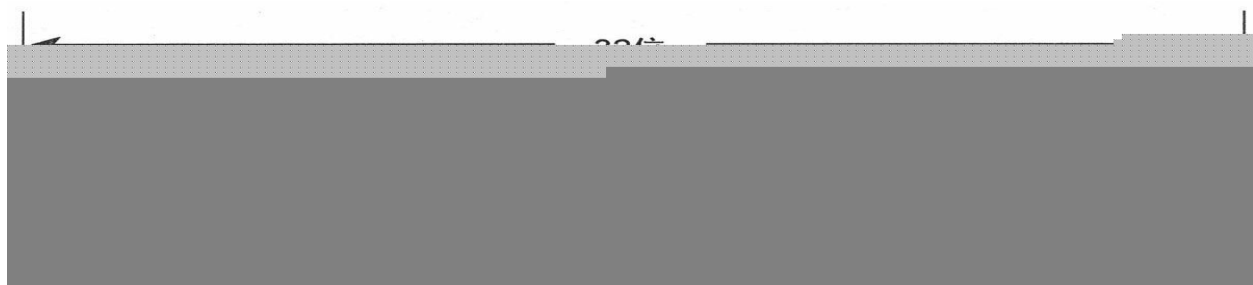
参见示例9-3，使用命令**ipv6 ospf area** 在路由器Hedwig的串行接口Serial0/0与以太网接口Ethemet0/0上启用了OSPFv3协议，并把路由器Hedwig的串行接口配置到区域1，把以太网接口配置到区域0中，同时在该路由器上创建了一个ID为“1”的OSPFv3进程。在OSPFv2中，需要两个命令才能完成相同的任务：使用**router ospf 1** 命令创建一个OSPF进程，然后在接口上使用**network area** 命令启用OSPFv2协议。这里有一点要注意，虽然在多个接口上启用OSPFv2可以使用单条**network area** 命令，但是在OSPFv3中必须在每一个需要运行OSPFv3协议的接口上配置**ipv6 ospf area** 命令。

示例9-3 命令**show ipv6 protocol**显示了该路由器上IPv6的OSPF进程ID和配置到每一个区域中的接口



一个接口上所有的IPv6地址都属于该接口上创建的OSPF进程。例如，在示例9-4中，路由器Hedwig的以太网接口增加了辅助的IPv6地址。

示例9-4 路由器Hedwig的以太网接口Ethernet0/0上增加了一个辅助的IPv6地址。由于在这个接口上配置了OSPFv3协议，因此该接口上的这两个IPv6地址都会包含在OSPFv3进程中

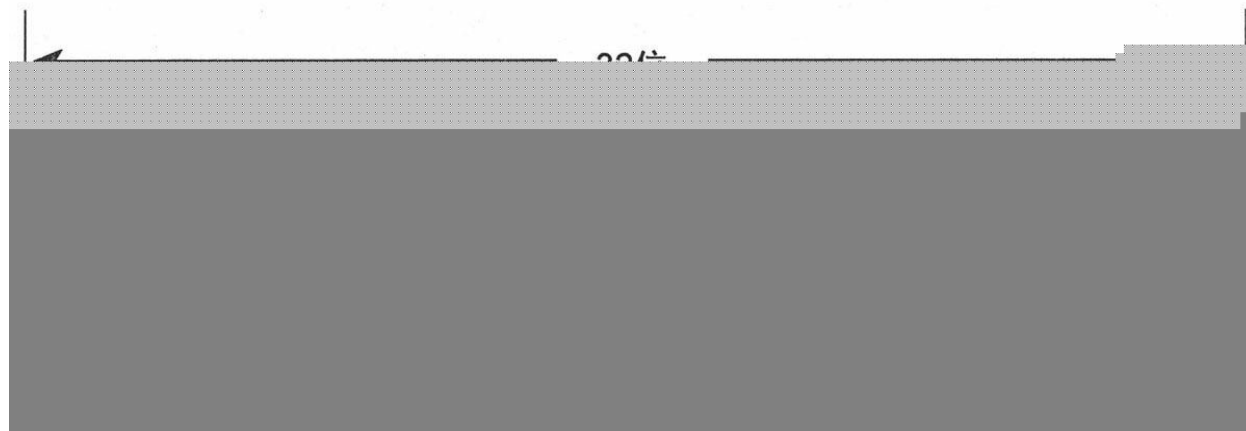


由于这个接口配置了OSPFv3协议，所以它的辅助地址将会自动地加入到OSPFv3的进程中去，而不需要额外的OSPFv3命令来为路由选择进程增加新的地址块。接口上的IPv6地址不能有选择地加入到OSPFv3进程中。它要么在接口上配置OSPFv3使接口的所有地址都加入到OSPFv3进程；要么在接口上不配置OSPFv3，从而没有任何地址加入到OSPFv3进

程。

OSPFv3路由器使用它们的链路本地地址作为Hello数据包的源地址。在Hello数据包中没有包含IPv6前缀的信息。在一条链路上能够配置多个IPv6地址。在单条链路上配置多个地址与IPv4中的配置不同，没有地址需要定义成辅助地址。在邻居之间除了链路本地地址外，即使它们没有共同的IPv6前缀，它们也可以建立邻接关系。这是OSPFv3与IPv4协议中的OSPFv2之间的不同之处。OSPFv2邻居之间只有属于相同的IP子网，并且这个共同的子网都配置作为邻居接口的主IP地址时，它们之间才能形成邻接关系。如图9-15所示，路由器Hedwig在它的以太网接口上配置了2001:db8:0:4::/64和2001:db8:0:5::/64。路由器Pigwidgeon的以太网接口配置了2001:db8:0:5::/64，路由器Crookshanks的以太网接口配置了2001:db8:0:4::/64。在示例9-5中，显示了Crookshanks同时与Hedwig（10.1.1.1）和Pigwidgeon（10.1.3.1）建立了邻接关系。其中，路由器Pigwidgeon是备份指定路由器。

示例9-5 使用**show ipv6 ospf neighbor**显示了路由器**Crookshanks**的邻居是**FULL**状态（即完全邻接状态）的，尽管除了链路本地地址之外它们并没有共同的**IPv6**前缀



两个邻居之间在建立起邻接关系之前，其他的一些参数都必须匹配。这些参数和IPv4中的参数是相同的：邻居之间必须使用相同的区域ID，它们必须具有相同的Hello时间间隔和无效时间（dead time），并且它们必须具有相同的E位数值用来标识区域是否是一个末梢区域。一个OSPFv3数据包也必须具有与接收接口相同的接口ID，否则这个OSPFv3数据包将被丢弃。在本章后面的9.2.3小节中将会讨论接口ID。

OSPFv3使用32位的数字作为路由器ID。如果路由器配置了IPv4，缺省情况下，选择RID的方式与OSPFv2中的方式相同。路由器loopback接口配置的最高IPv4地址将作为该路由器的RID。如果没有配置loopback接口，则选择所有其他接口上配置的最高地址作为它的RID。

IPv6邻居总是通过它们的RID进行告知。这与IPv4不同，在IPv4中，点到点网络的邻居是通过RID告知的，而广播网络、NBMA和点到多点网络的邻居都是通过它们的接口地址告知的。邻居的ID参见示例9-5，它显示了路由器的ID是通过配置的IPv4地址获得的。

如果在网络中没有配置IPv4协议地址，你也不希望仅仅因为得到一个路由器ID而配置IPv4地址，那么在启动OSPF进程之前，就必须使用IPv6的OSPF路由选择进程命令**router-id** 配置路由器ID。

当在接口上配置了OSPFv3后，路由选择进程就创建了。像链路的成本代价等接口参数可以在接口配置模式下更改，而全局参数则需要在OSPF进程模式下更改配置。

9.2.2 案例研究：末梢区域

命令**ipv6 router ospf** 可以让路由器切换到全局的进程配置模式，这和IPv4中的**router ospf** 命令是一样的。IPv6中的配置方式与IPv4中的配置也是一样的。在IPv6中，可以用与IPv4中OSPFv2完全相同的方式利用**area stub**、**area nssa**和**area stub no-summary** 命令来支持和配置末梢、NSSA和完全末梢等区域。在路由器Hedwig和Scabbers上利用示例9-6和示例9-7中的配置可以把图9-15中的区域1配置成一个完全末梢区域。

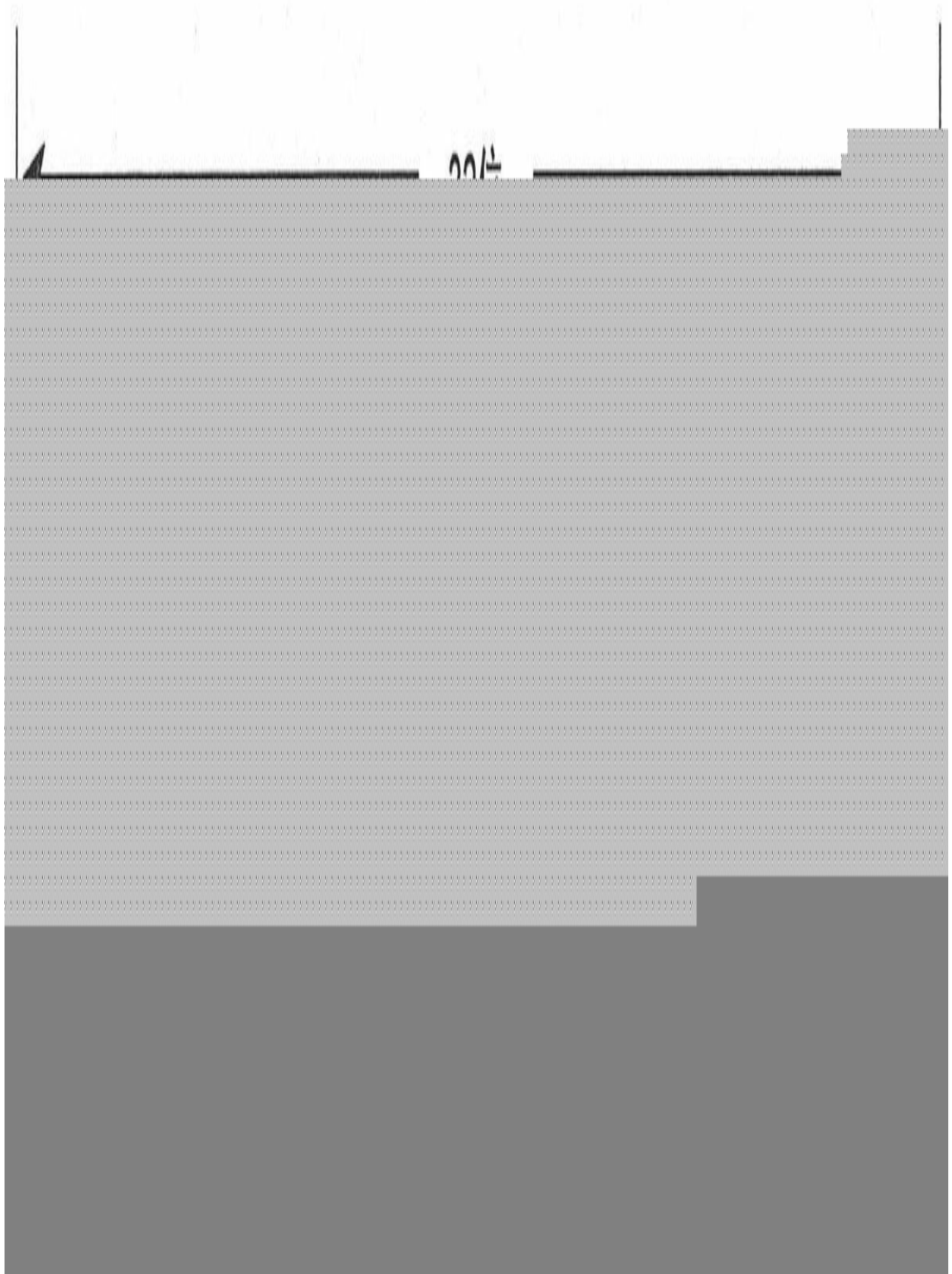
示例9-6 在路由器**Hedwig**上，区域**1**被配置成一个完全末梢区域，**no-summary**只能配置在ABR上

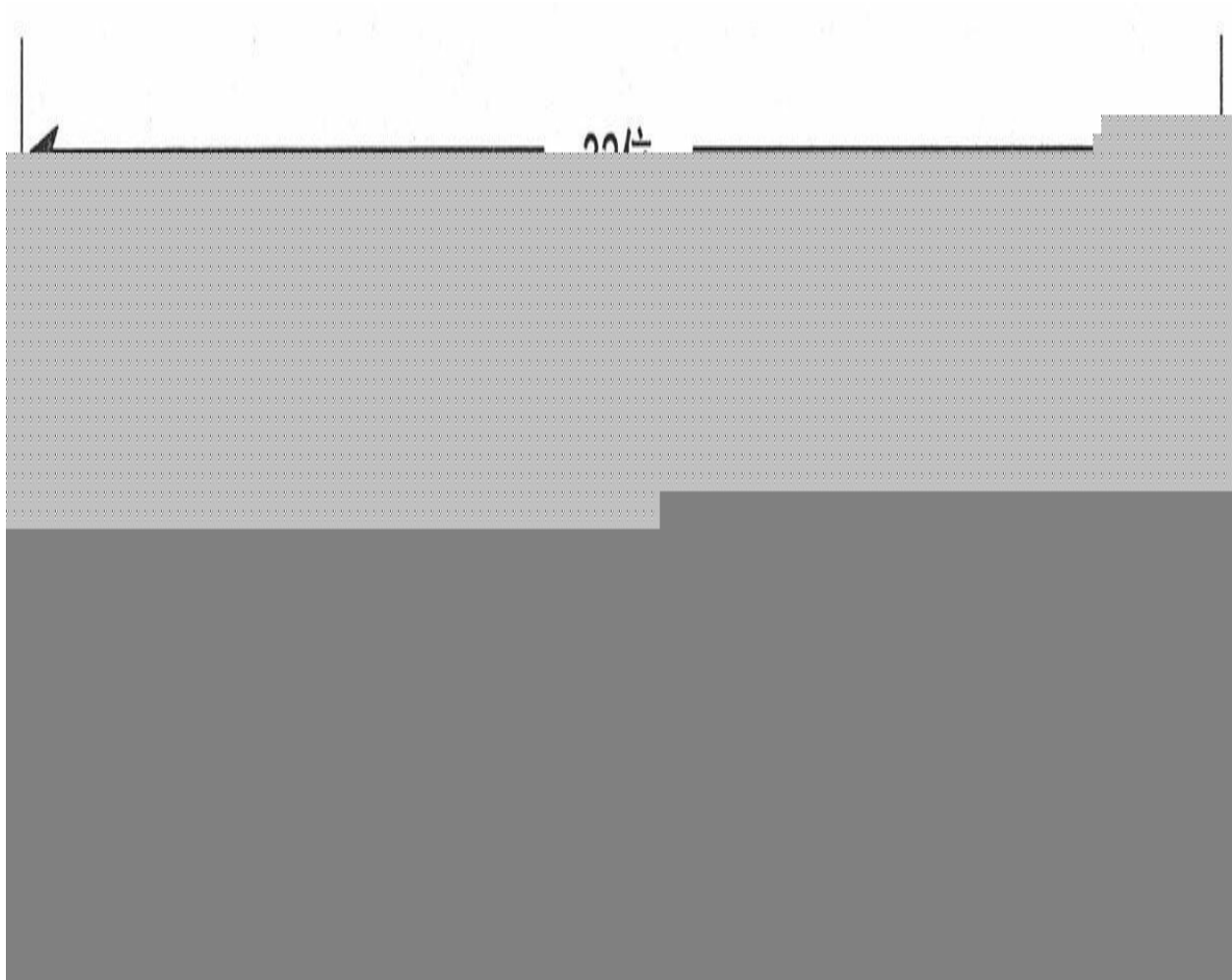


示例9-7 路由器**Scabbers**上的区域**1**也被配置成一个末梢区域，因为末梢区域中的所有路由器都必须被配置为一个末梢

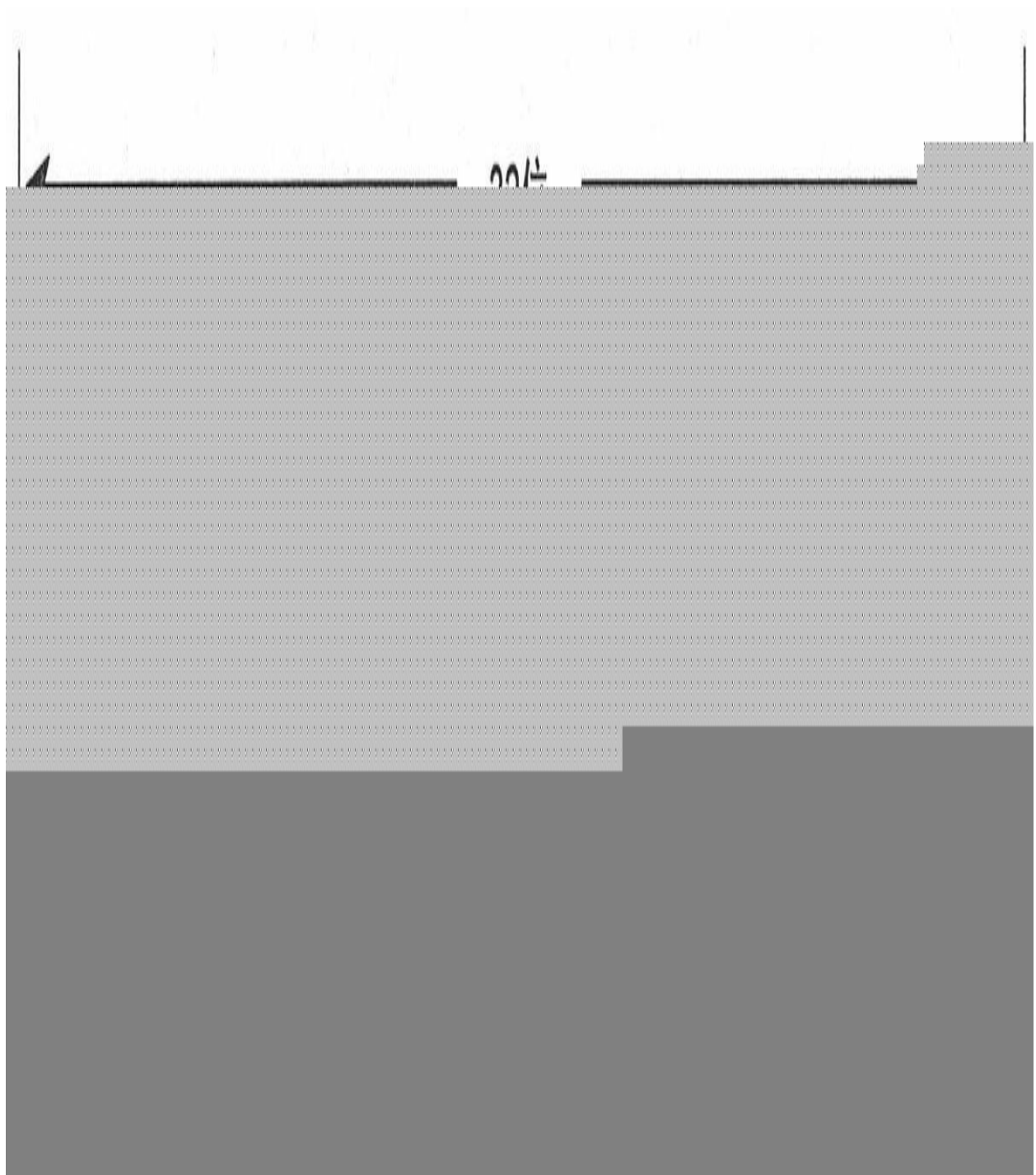
在示例9-8中，显示了路由器Scabbers的区域1变成完全末梢之前时它的IPv6路由表，而在示例9-9中显示了Scabbers作为完全末梢区域的节点后的路由表。

示例9-8 路由器Scabbers的IPv6路由表，它包含了**10条OSPF**条目，都是通过路由器Hedwig学习到的





示例**9-9** 作为完全末梢区域的一个节点，路由器**Scabbers**现在只有一条**OSPF**条目了，即缺省路由



9.2.3 案例研究：一条链路上的多个实例

在图9-16中，在以太网的网段上增加了一台新的路由器Hermes。假设我们希望实现的设计是把路由器Pigwidgeon与Hermes之间的OSPFv3通信量，和路由器Hedwig与Crookshanks之间的通信量隔离开来。根据现有

的配置，网络将会在4台路由器之间选择DR和BDR，并且每一台路由器都和DR与BDR建立邻接关系。OSPFv3的Hello数据包包含了一个实例ID。这个实例ID可以用来把运行在同一个LAN上的两个OSPF进程分开。所收到的Hello数据包中的实例ID必须与接收该数据包的路由器接口上配置的实例ID相匹配，否则该数据包将被丢弃。如果实例ID的值没有另外指定，那么缺省地使用实例ID 0。在路由器Pigwidgeon和Hermes配置与路由器Hedwig和Crookshanks不同的实例ID，就可以建立预期设计的邻接关系。

路由器Pigwidgeon的配置更改如示例9-10所显示的那样。

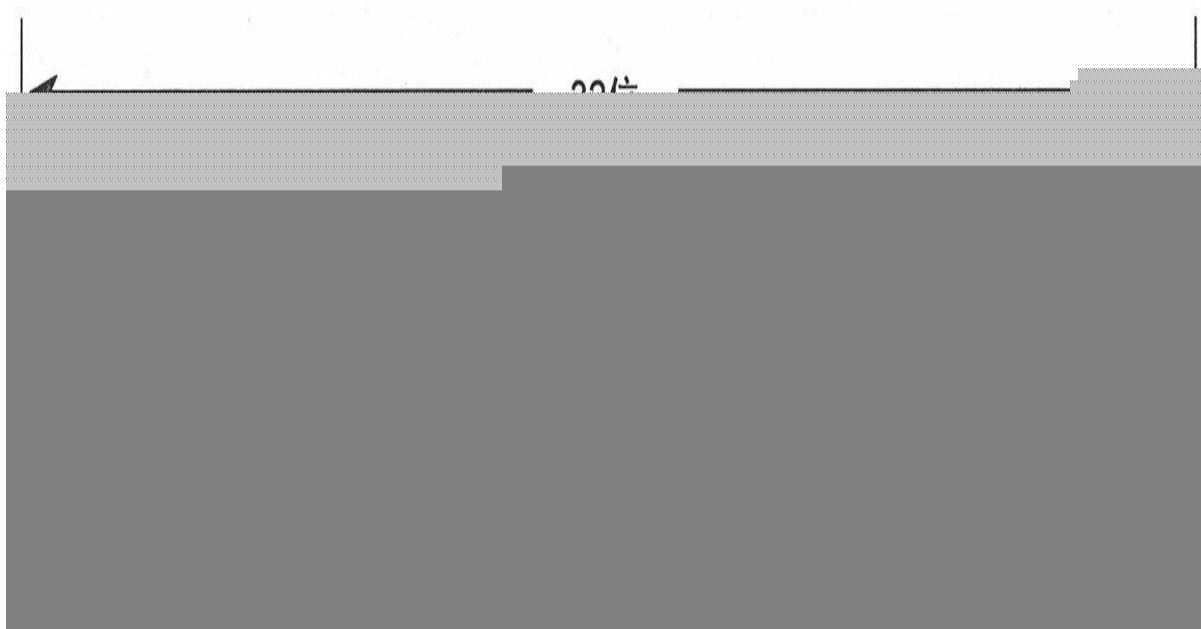
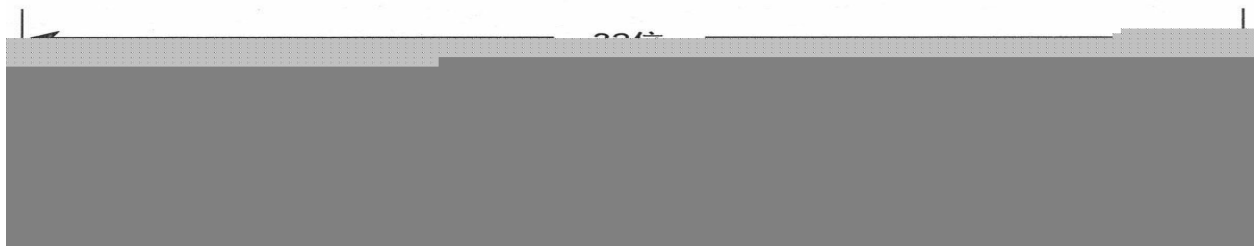


图9-16 在图9-15的网络中增加一台新的路由器

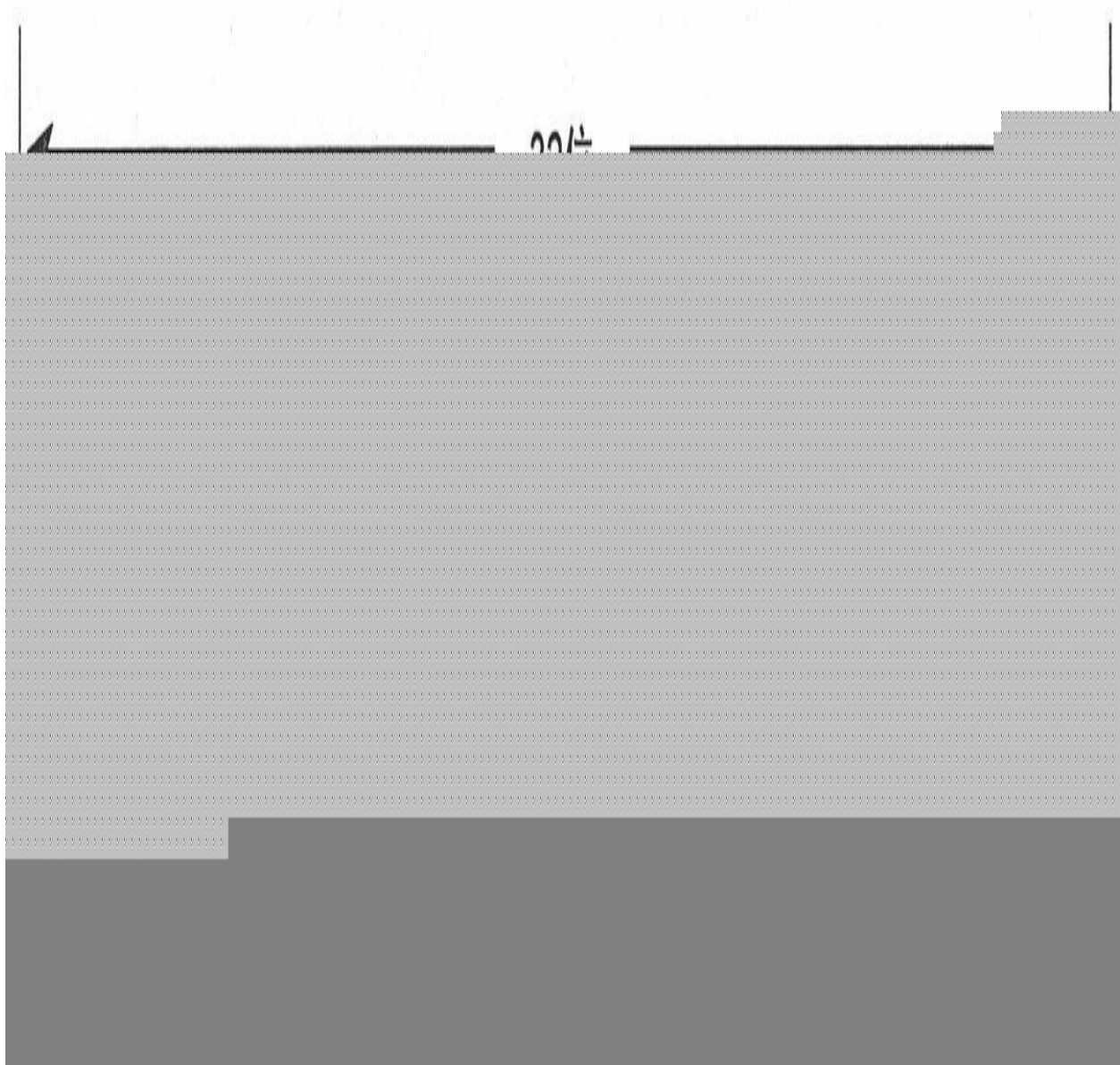


实例9-10 路由器Pigwidgeon的实例ID由缺省的实例ID 0更改为1，这样在以太网上创建了一个明确的OSPF进程

路由器Pigwidgeon的实例ID由缺省的0更改为1。路由器Hedwig和

Crookshanks继续使用缺省值0作为它们的实例ID。可以使用与路由器Pigwidgeon相似的配置将路由器Hermes的实例ID改为1。参见示例9-11，可以看到路由器Hermes的Ethernet0/0接口运行的IPv6 OSPF配置，其中只有路由器Pigwidgeon（10.1.3.1）是建立邻接的。

虽然在一台路由器上可以运行多个OSPFv3进程，但是在一个接口上只能运行单个进程或实例。



实例9-11 路由器Hermes的IPv6 OSPF以太网接口的配置显示了Pigwidgeon是建立邻接的，因为路由器Pigwidgeon是这个以太网段上除

Hermes外惟一使用实例ID为1的路由器

9.2.4 案例研究：运行在NBMA网络上的OSPFv3配置

对于NBMA网络，OSPFv3和OSPFv2的配置选项是相同的。也就是说，从OSPF的角度来看，NBMA网络可以保留NBMA的配置，或者当作广播网络或点到多点的网络配置，或者也可以当作使用子接口配置的点到点网络配置。点到点链路可以直接进行配置，其配置方法与通过串行链路直接相连的路由器的配置相同。在9.2.1小节中的路由器Hedwig和Scabbers就是配置成直接相连的串行链路。如果这两台路由器是通过点到点的帧中继子接口相连的，那么相应的OSPFv3的配置应该是一样的。在本小节中将会讲述NBMA、广播网和点到多点网络。

在OSPFv3能够学习到相关的邻居之前，OSPF用到的邻居地址必须先与穿过NBMA网络的特定虚电路相关联。建立帧中继映射来标识远程的IPv6地址映射到与本地路由器相连的某条PVC电路上^[2]。由于IPv6可以在单个接口上配置多个地址，那么对于每一条PVC电路都需要多个map语句。所有的OSPFv3数据包都使用链路本地地址作为源地址。对于单播的OSPFv3数据包来说，也使用链路本地地址作为目的地址和在路由选择转发时的下一跳地址。因此，至少链路本地地址必须被映射。如图9-17所示，图中显示了一个由4台相连的路由器组成的帧中继网络。

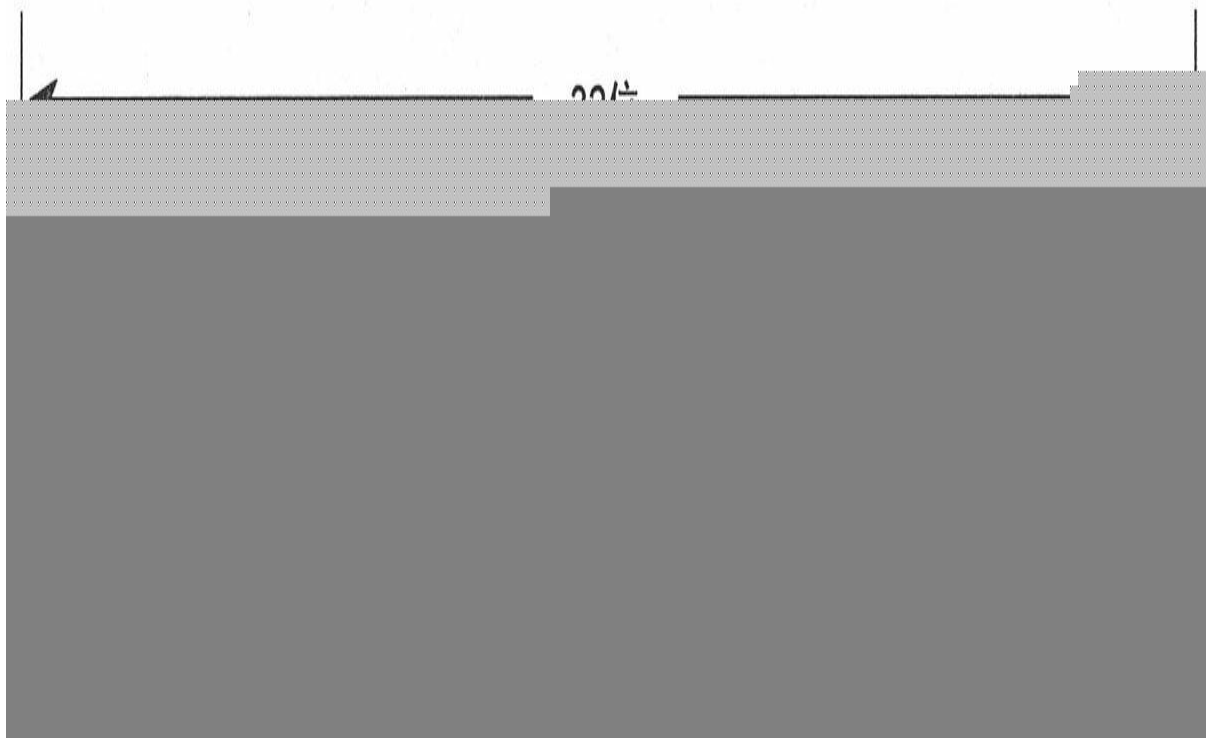
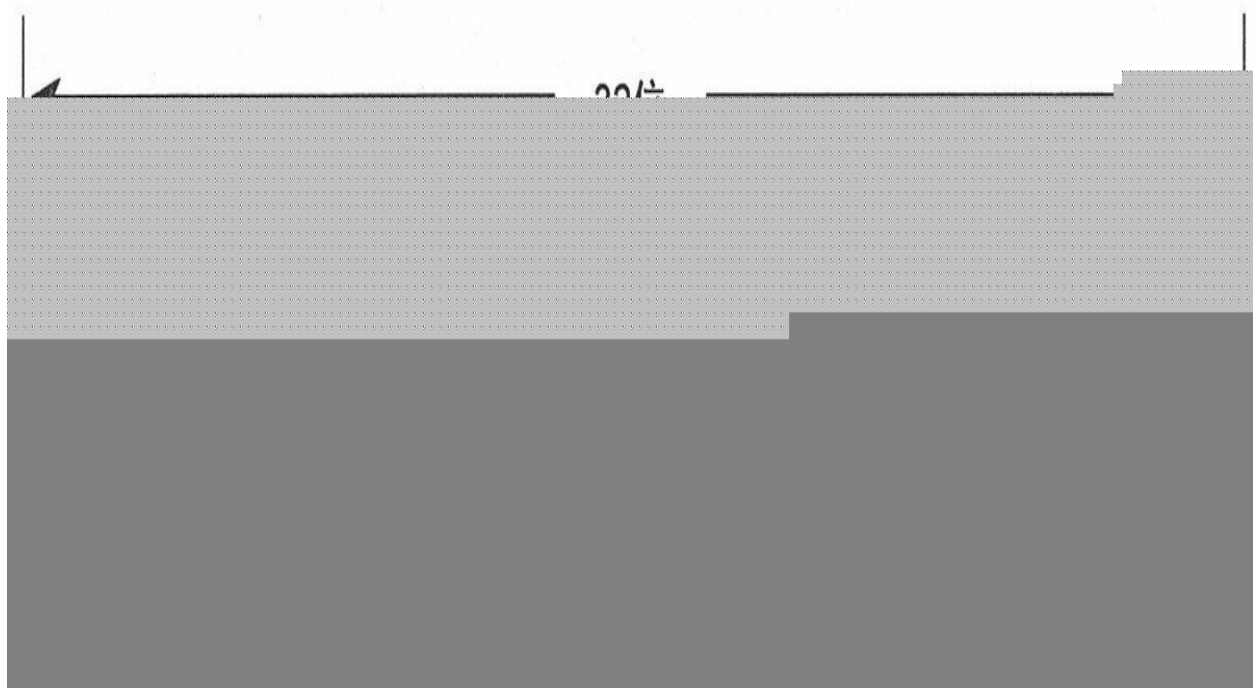


图9-17 在这个IPv6的帧中继网络上，配置OSPFv3的几个选项

配置OSPFv3路由器的第一种方法是手工配置OSPFv3邻居。首先远程路由器的链路本地地址映射到正确的DLCI上，接着定义IPv6 OSPFv3的邻居，这两个步骤都是在接口配置模式下完成的。示例9-12中显示了路由器Skrewt的配置。

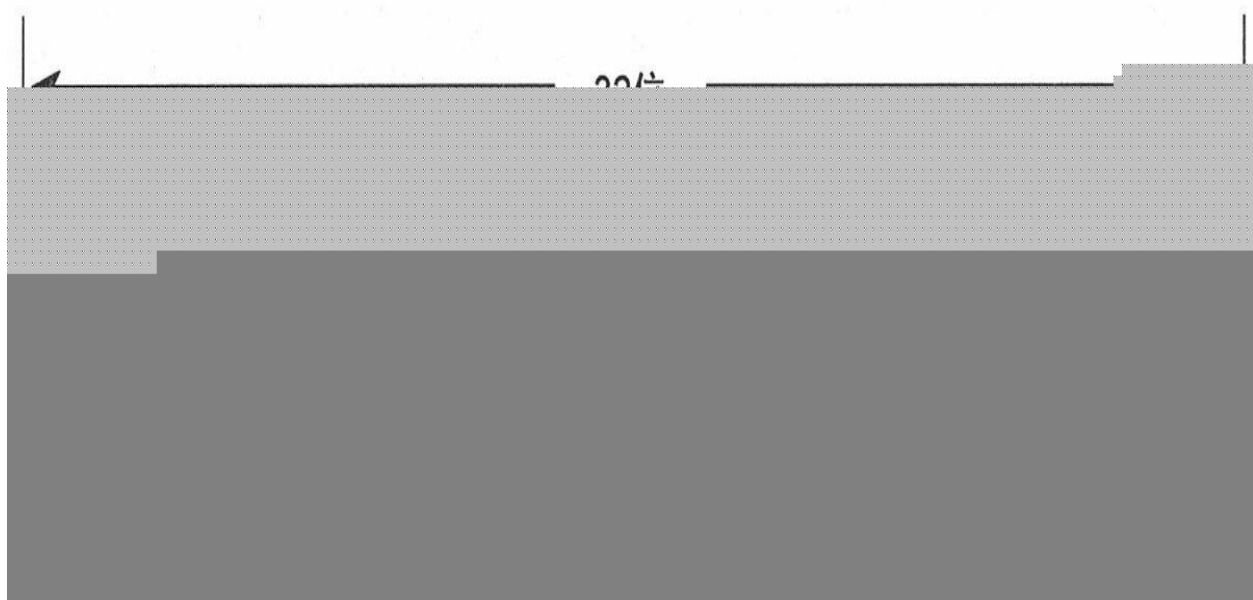
示例9-12 在路由器**Skrewt**上手工配置**OSPFv3**邻居



这里请注意，在**frame-relay map** 语句和**ospf neighbor** 语句中使用的是本地链路地址。通过在命令**ipv6 ospf neighbor** 中使用可选的关键字可以指定优先级，以便指出哪一台路由器将成为DR和BDR。

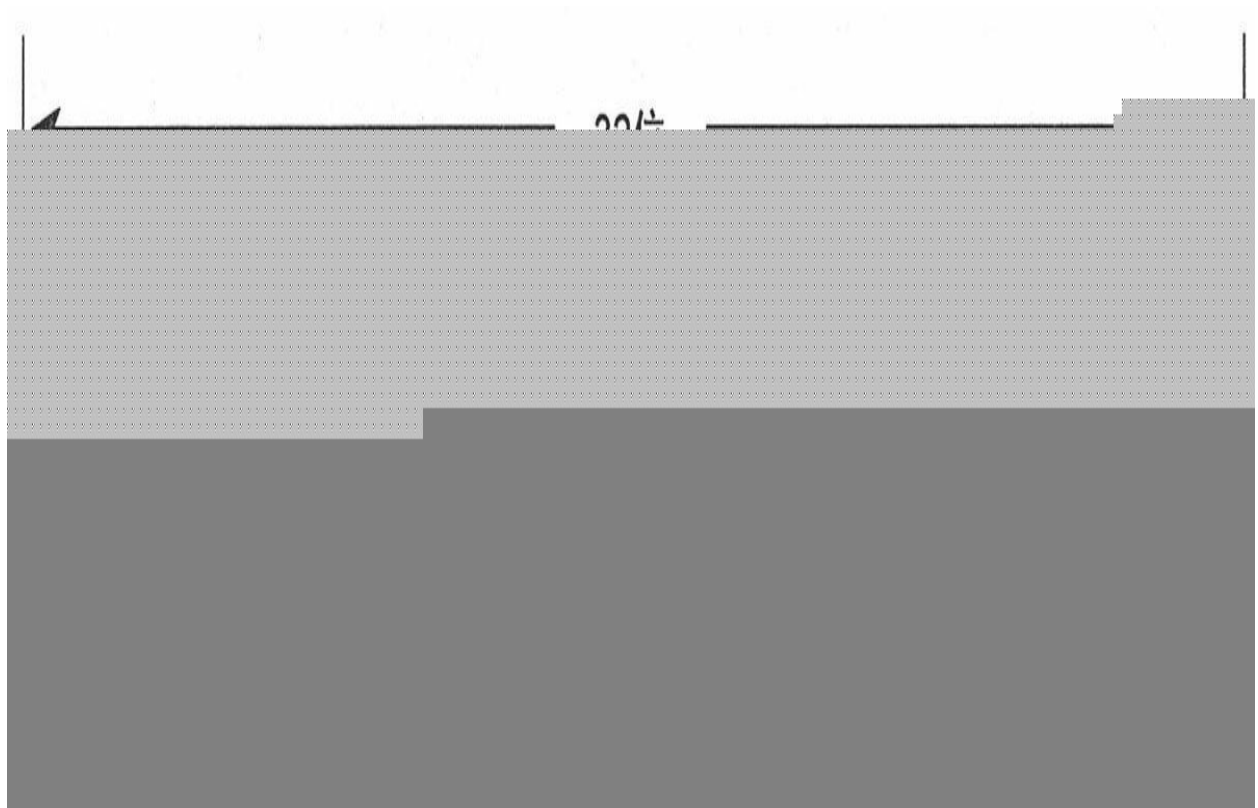
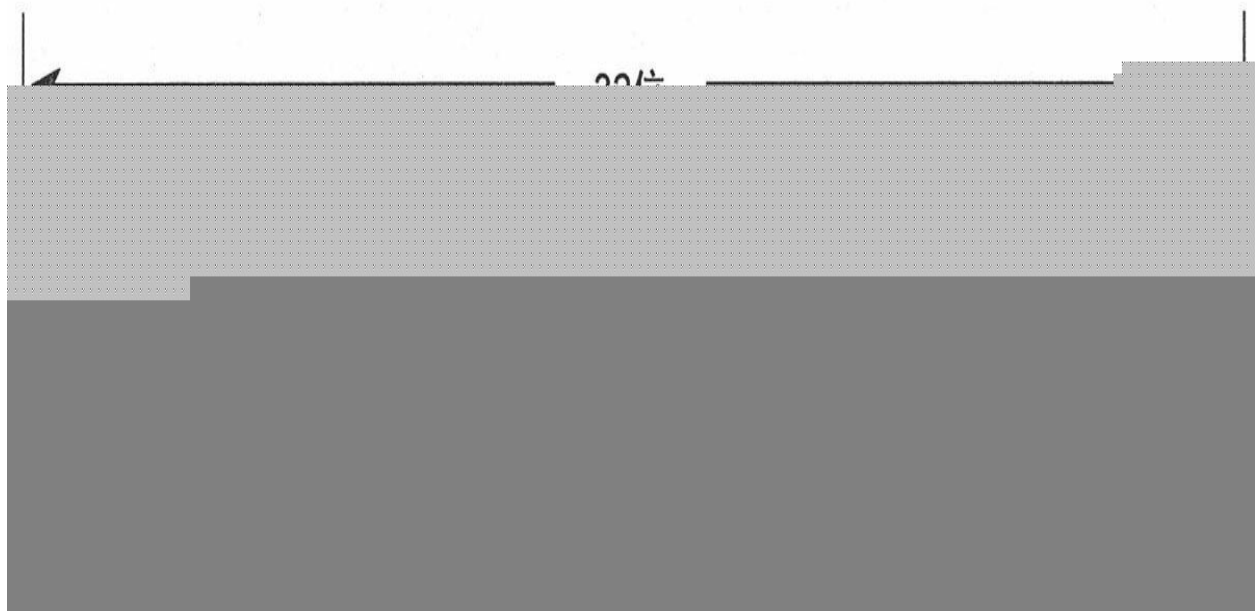
配置OSPFv3路由器的另一种方法是，OSPFv3动态地发现它的邻居。这种配置方法需要两个步骤：首先使用**ipv6 ospf network broadcast** 命令将一个OSPF网络定义成广播网络；然后在**frame-relay map** 语句使用关键字**broadcast** 指出在PVC电路上进行广播转发。路由器Skrewt的配置更改参见示例9-13。

示例9-13 路由器Skrewt被配置为广播型的PVC电路，运行在帧中继链路上的OSPFv3网络也被配置成为一个广播型网络



在上面多路访问网络中其他路由器的配置也是类似的。但注意不再有任何需要定义IPv6 OSPF邻居了。如果像下面的NBMA网络那样并不是全连接的网络结构（每一台路由器与其他的任何一台路由器之间都有PVC电路相连），那么DR和BDR的选择必须谨慎。DR和BDR路由器和其他的任何一台路由器都必须具有完整的虚电路连接，只有这样才能建立起正确的邻接关系。利用接口配置模式命令**ipv6 ospf priority**可以在你希望作为DR的路由器上指定一个较高的优先级。指定DR的方法我们已经在8.2.12小节中讲述。参见示例9-14，显示了路由器Skrewt的串行接口Serial 0/0的IPv6 OSPF配置。

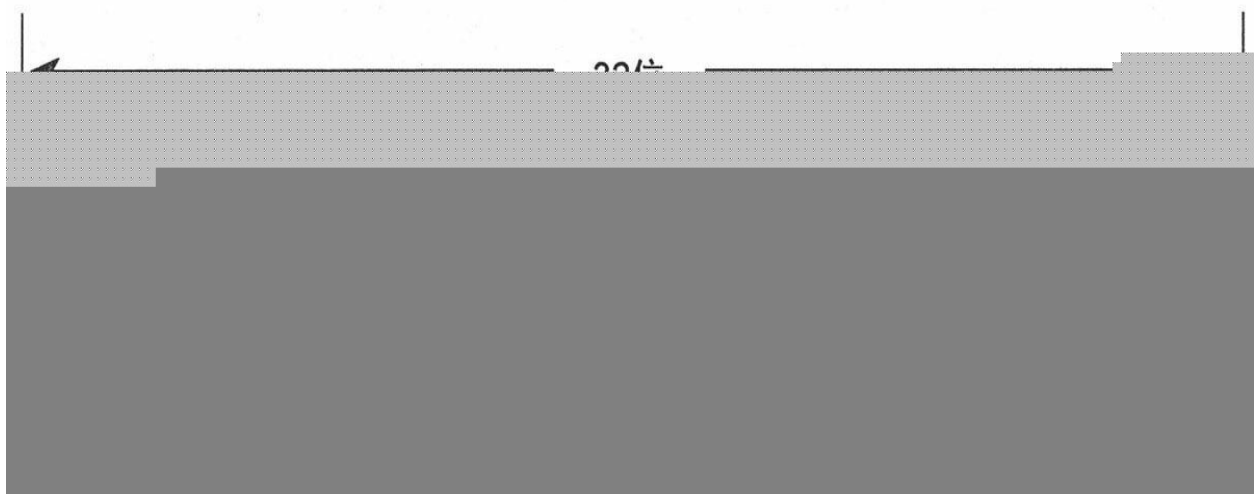
示例9-14 使用**show ipv6 ospf interface**命令可以显示路由器接口的IPv6 OSPFv3配置



请注意，这里的网络类型是**BROADCAST**（广播型）。读者也可以注意到，它的邻居数目是3个，并存在3个邻接关系。这台路由器是**DR**路由器，因此它和网络上其他所有的路由器都建立了邻接关系。那些不是**DR**或**BDR**的邻居也具有3个邻居，但只有两个邻接关系，这是因为广播

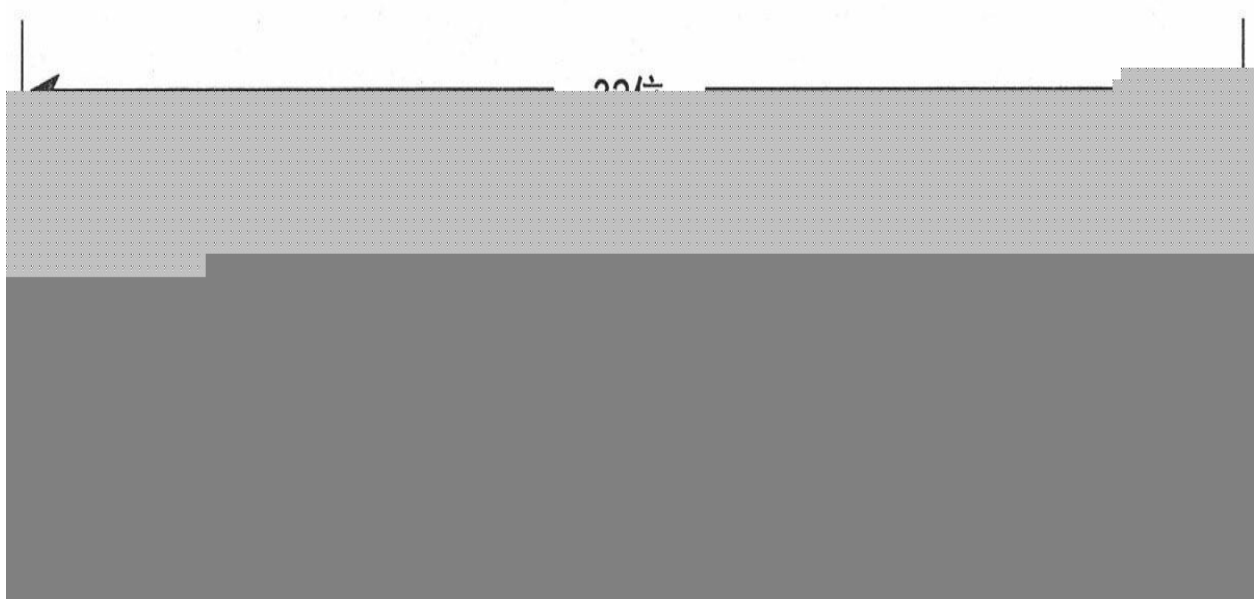
型网络上的路由器只需要与DR和BDR之间建立邻接关系。例如路由器Flobberworm，参见示例9-15。

示例9-15 在广播型网络上的4台路由器中，不是**DR**或**BDR**的路由器将拥有**3**个邻居，但只具有两个完全的邻接关系



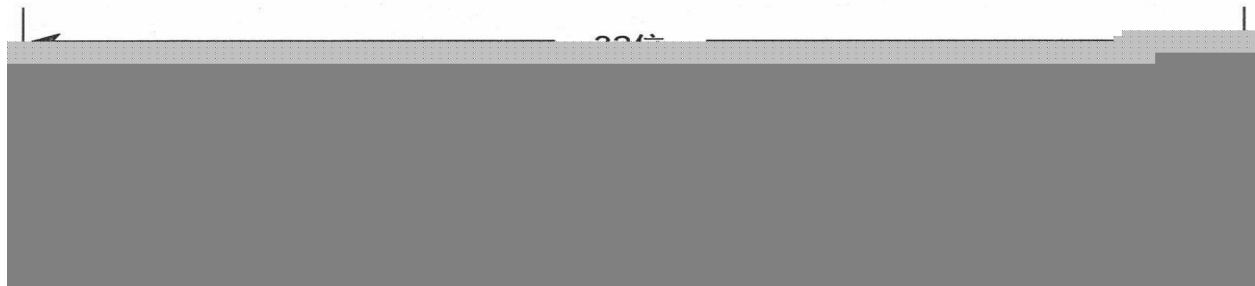
为了避免进行DR/BDR的选举，可以将OSPF网络的类型改为点到多点的网络类型。参见示例9-16，显示了路由器Skrewt更改后的配置。

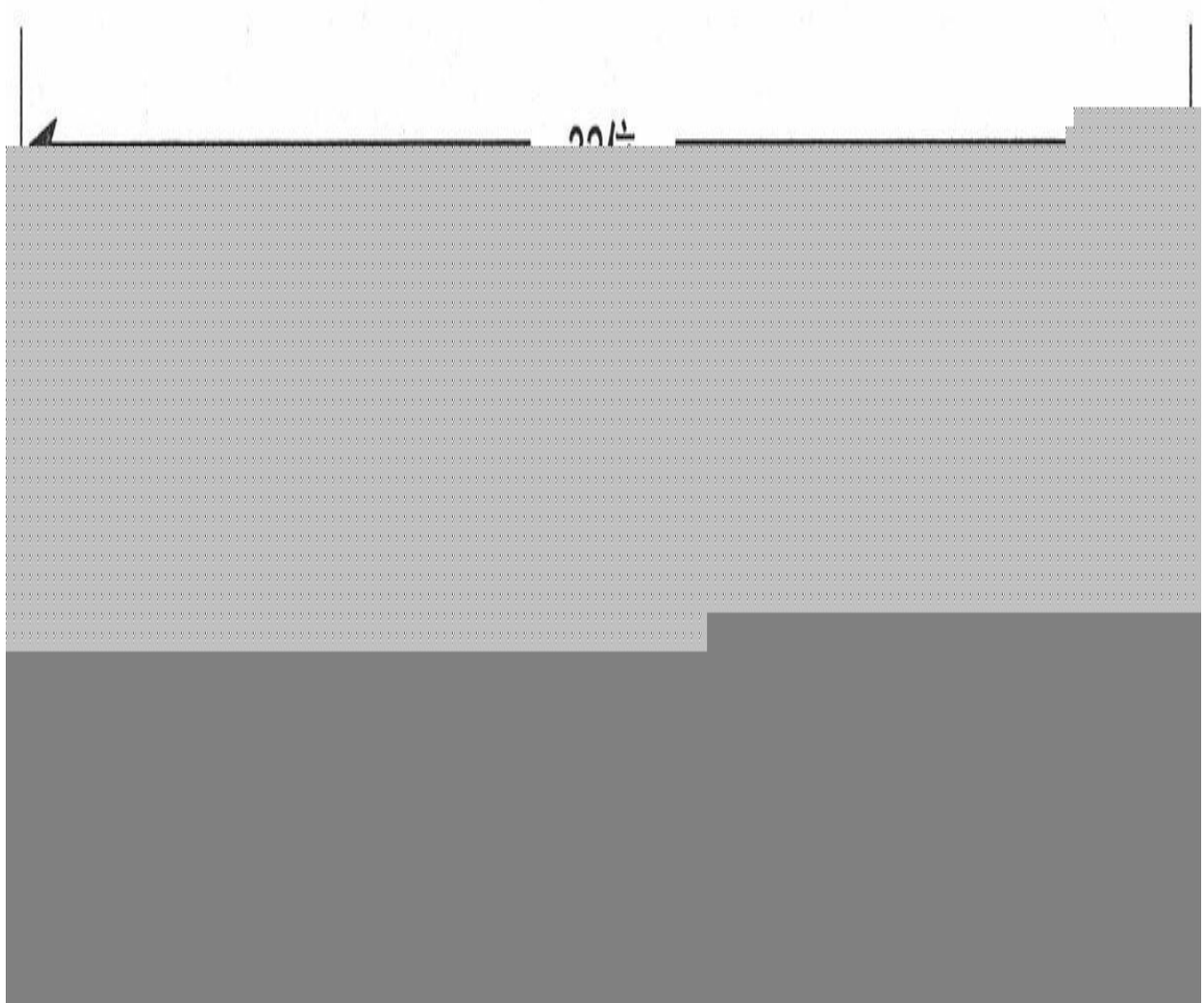
示例9-16 路由器Skrewt的配置显示PVC电路仍然为广播型的，但**OSPFv3**的网络类型已经改变为点到多点类型



参见示例9-17，路由器Skrewt的接口配置显示了点到多点网络的缺省计时器与广播型网络的缺省计时器有所不同。点到多点网络上的Hello数据包是每30s发送一次，而广播型网络上的Hello数据包是每10s发送一次。另外，OSPFv3接口配置没有任何DR，路由器和所有的3台邻居路由器都建立了邻接关系。

示例9-17 使用**show ipv6 ospf interface**命令可以显示一个**OSPF**点到多点网络上**IPv6 OSPF**接口的配置





将一个NBMA网络配置成OSPF点到多点网络不需要PVC电路具有全连接。如图9-18所示，图中显示了一些PVC电路已经从图9-17的网络中去掉。在示例9-18中，配置DLCI 202的**map** 语句也从路由器Skrewt上去掉了。

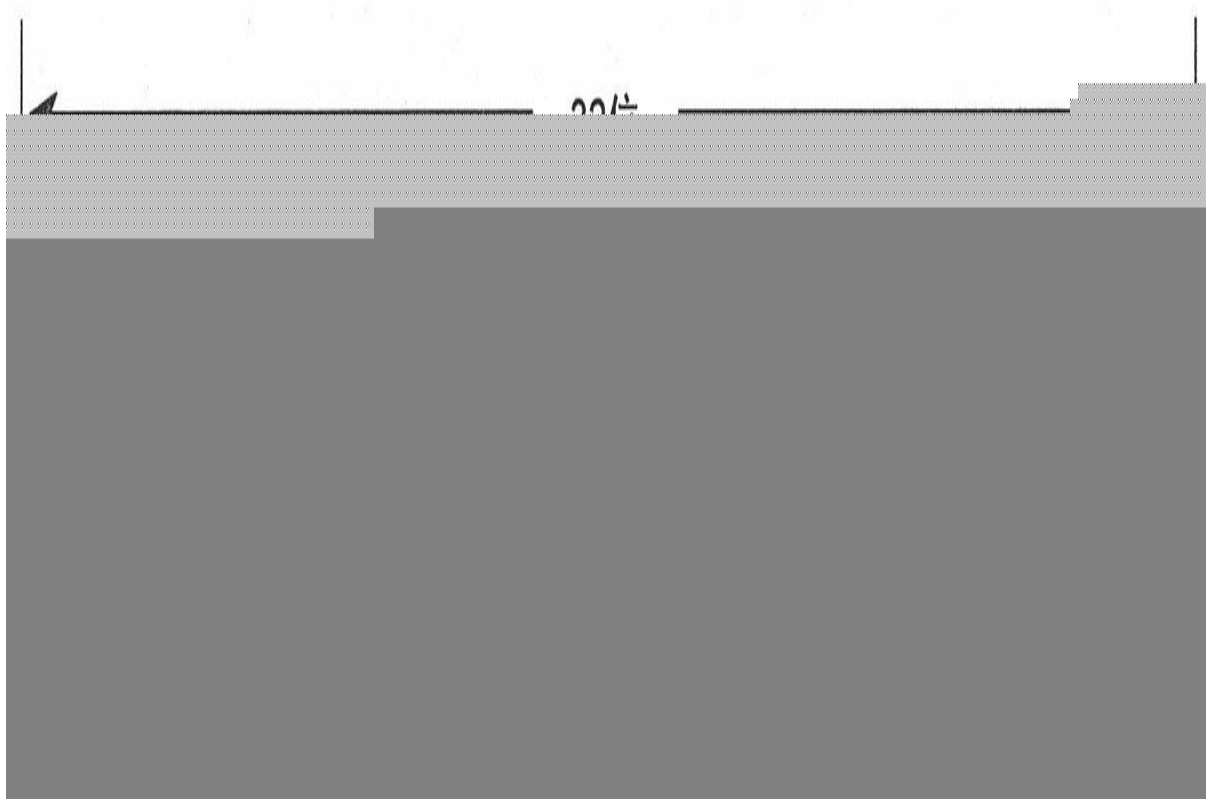
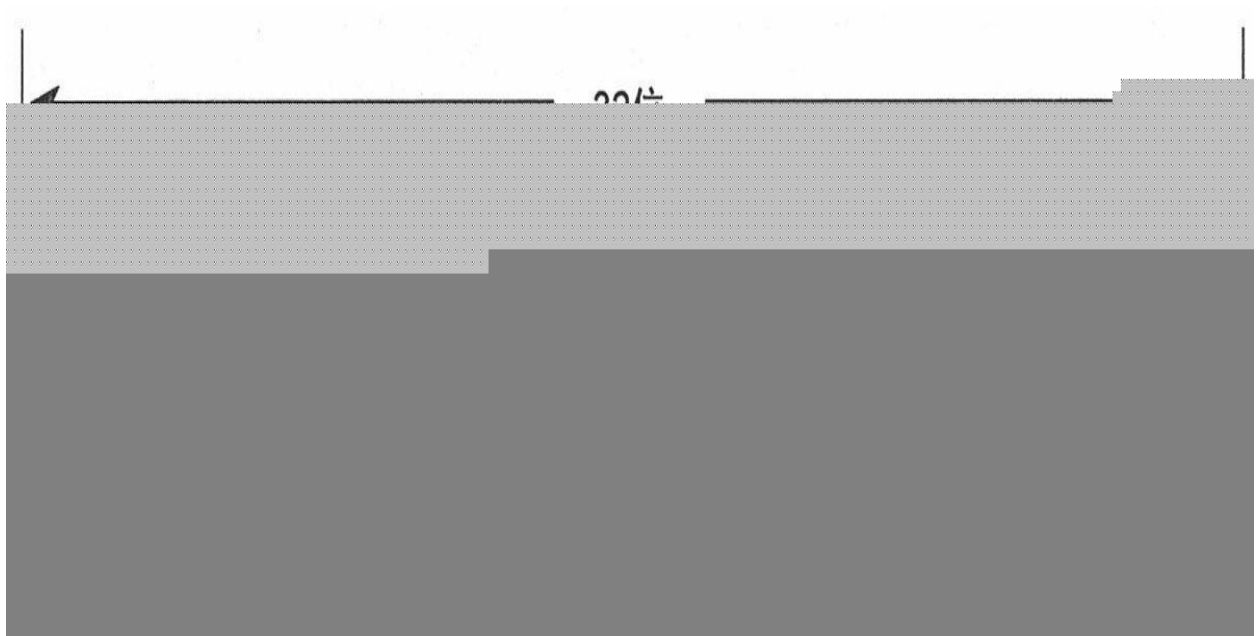


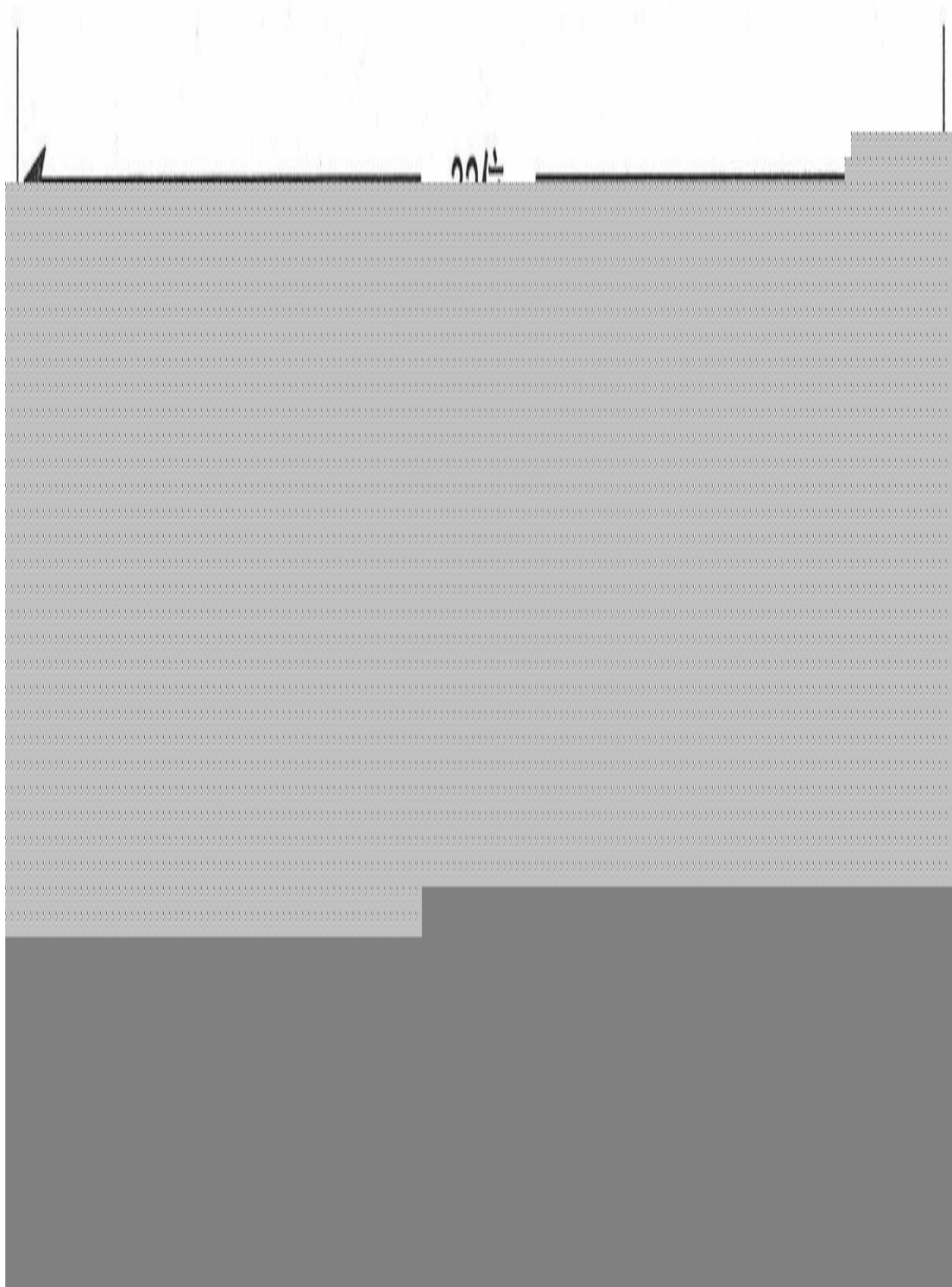
图9-18 去掉一些PVC电路的图9-17的网络

示例9-18 路由器Skrewt的帧中继配置映射IPv6地址到剩下的两条PVC电路上



正如示例9-19中所显示的，在路由器Skrewt的IPv6路由表中可以看到，对于Skrewt来说，路由器Hippogriff和它所学到的IPv6前缀都是可访问的。路由器Skrewt和Hippogriff都与路由器Thestral和Flobberworm形成了邻接关系。除了由Hippogriff通告的IPv6前缀，Hippogriff的串行接口Serial0/0的IPv6地址（2001:db8:0:1::3）是通过路由器Skrewt的串行接口Serial0/0学习到的，它的下一跳链路本地地址是FE80::206:28FF:FEB6:5BC0和FE80::201:42FF:FE79:E500。

示例9-19 路由器Skrewt的路由表显示，通过在点到多点网络上与路由器Skrewt有PVC电路连接，路由器Skrewt仍然可以访问那些与它没有PVC连接的路由器



配置在帧中继点到多点接口上的IPv6地址2001:db8:0:1::2/64、2001:db8:0:1::3/64和2001:db8:0:1::4/64，都是通过带有128位前缀长度的OSPF通告的。这些地址与路由器Hippogriff通告到OSPF里的其他IPv6前缀地址，都可以通过路由器Skrewt的两台邻接路由器的链路本地地址到达。

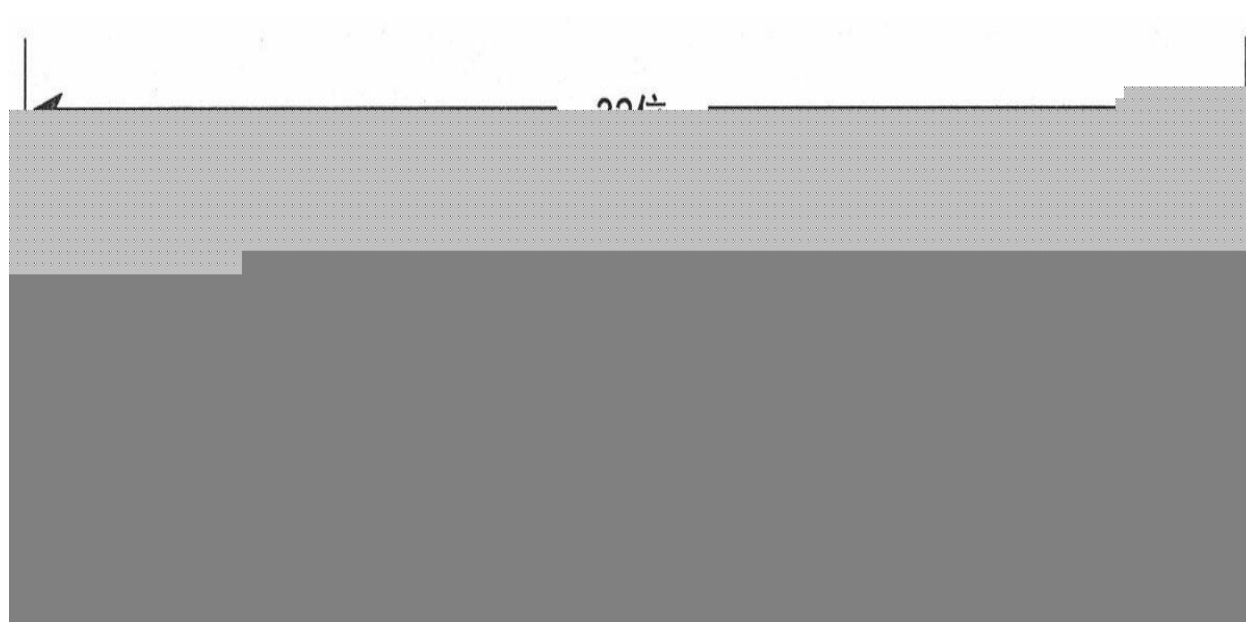
9.3 OSPFv3的故障诊断

IPv6协议的OSPFv3的故障诊断所使用的方法实际上与IPv4协议中的OSPFv2是相同的。主要的不同之处是寻址：直接发送数据包到一个邻居时，OSPFv3使用链路本地地址作为数据包的源地址和目的地址。

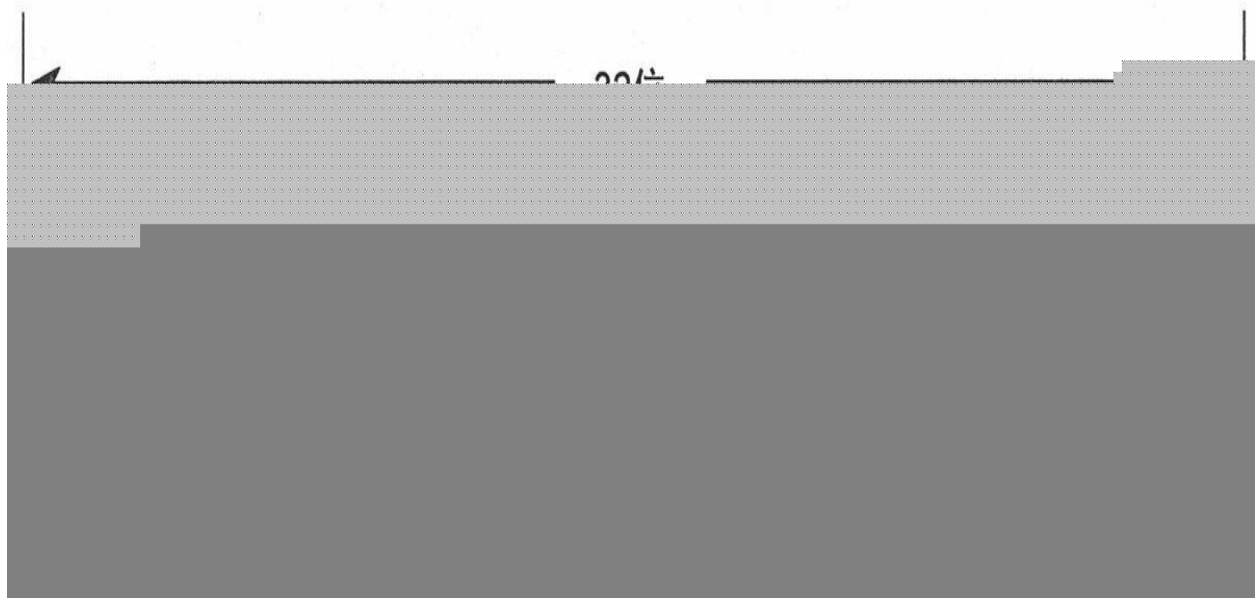
案例研究：帧中继的映射

回到上面图9-17的最初配置，路由器Skrewt和Hippogriff配置参见示例9-20和示例9-21。

示例9-20 路由器Skrewt最初帧中继映射的配置



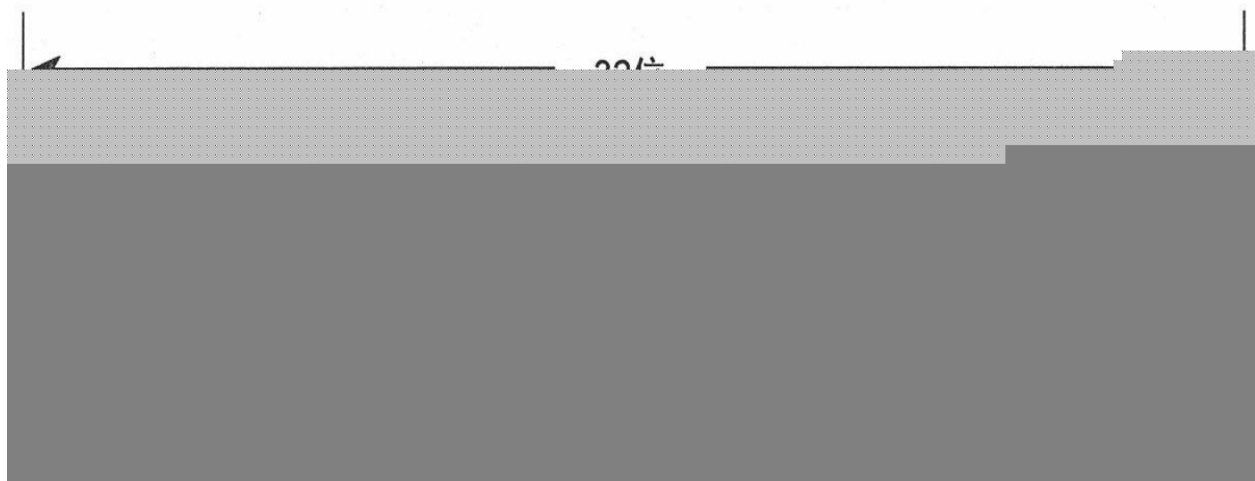
示例9-21 路由器Hippogriff最初帧中继映射的配置



其他路由器的配置也是相似的。

参见示例9-22，显示出路由器Skrewt的OSPFv3邻居表中路由器之间没有形成邻接关系。

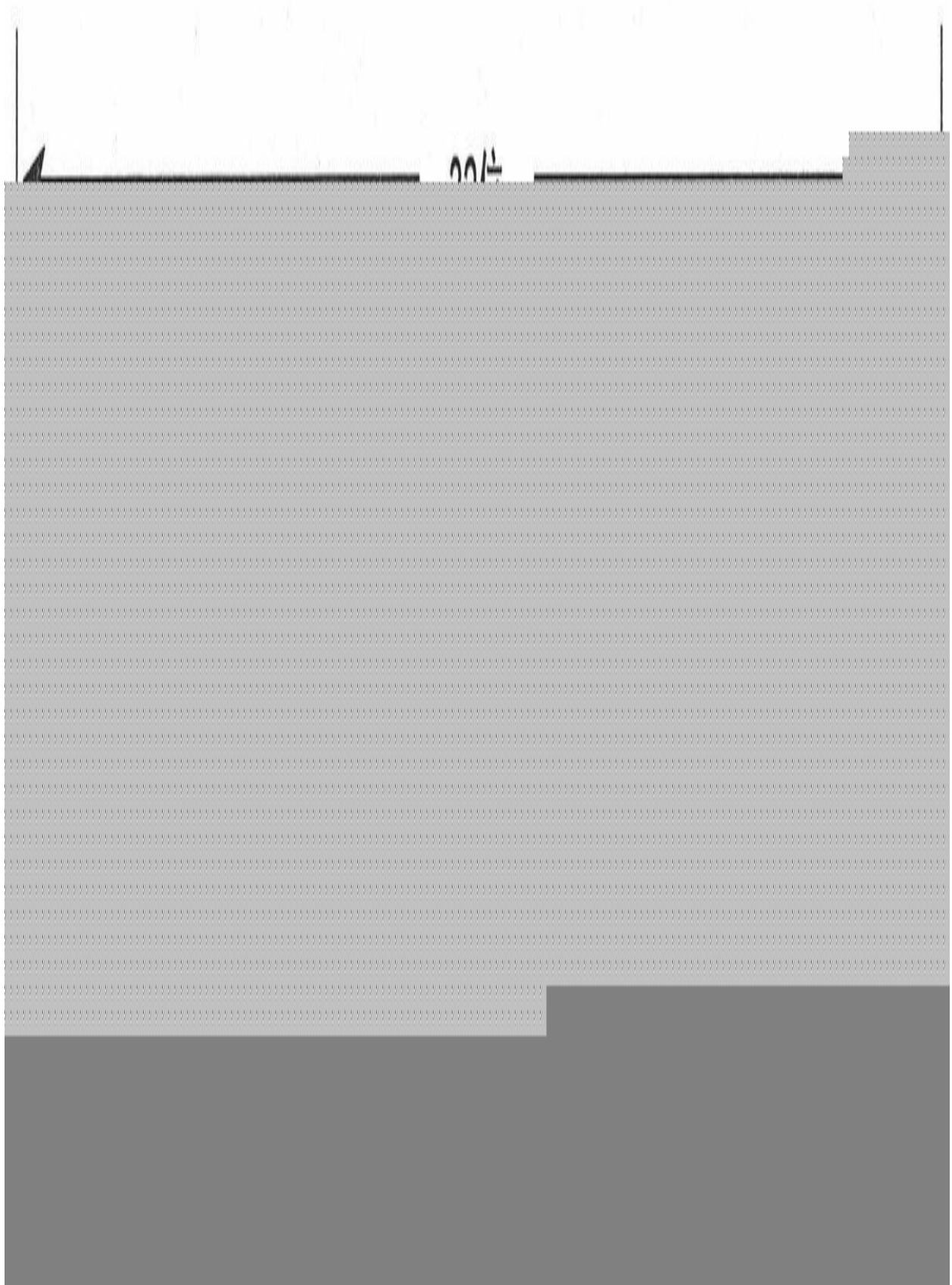
示例9-22 路由器Skrewt没有和DR与BDR建立邻接关系



参见示例9-23所示，对IPv6 OSPF的Hello数据包与邻接关系进行调试，显示出可以收到Hello数据包，并建立了双向通信状态。同时可以看到也选举出了DR和BDR路由器。数据库同步尝试完成与DR和BDR之间的邻接。虽然路由器Skrewt能够连续地收到来自可能的DR路由器的Hello数据包，但是它尝试发送数据库描述数据包（DBD）到可能的DR路由

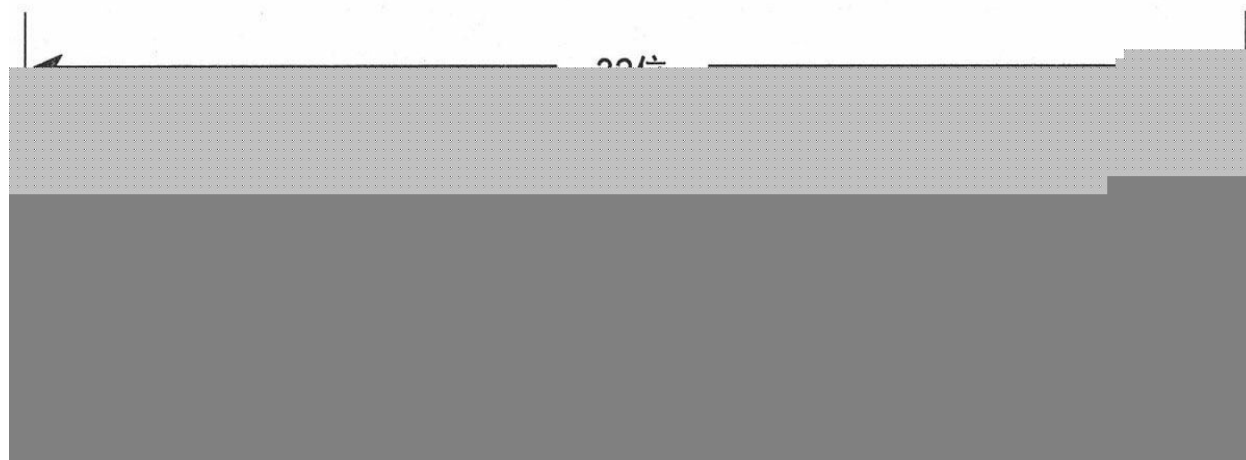
器时却没有收到任何确认。与DR和BDR路由器之间的邻居状态是EXSTART，这表明正在建立主 / 从关联关系，并已经发送了一个初始的DBD数据包。

示例9-23 使用命令**debug ipv6 hello**和**debug ipv6 ospf adj**显示了Hello数据包、双向通信连接建立、**DR/BDR**选举，以及正在发送**DBD**数据包



读者在这里注意到并没有收到确认数据包。进一步使用有关IPv6 OSPF数据包的调试命令，来显示有关正在发送的DBD数据包更多的信息^[3]，参见示例9-24所示。

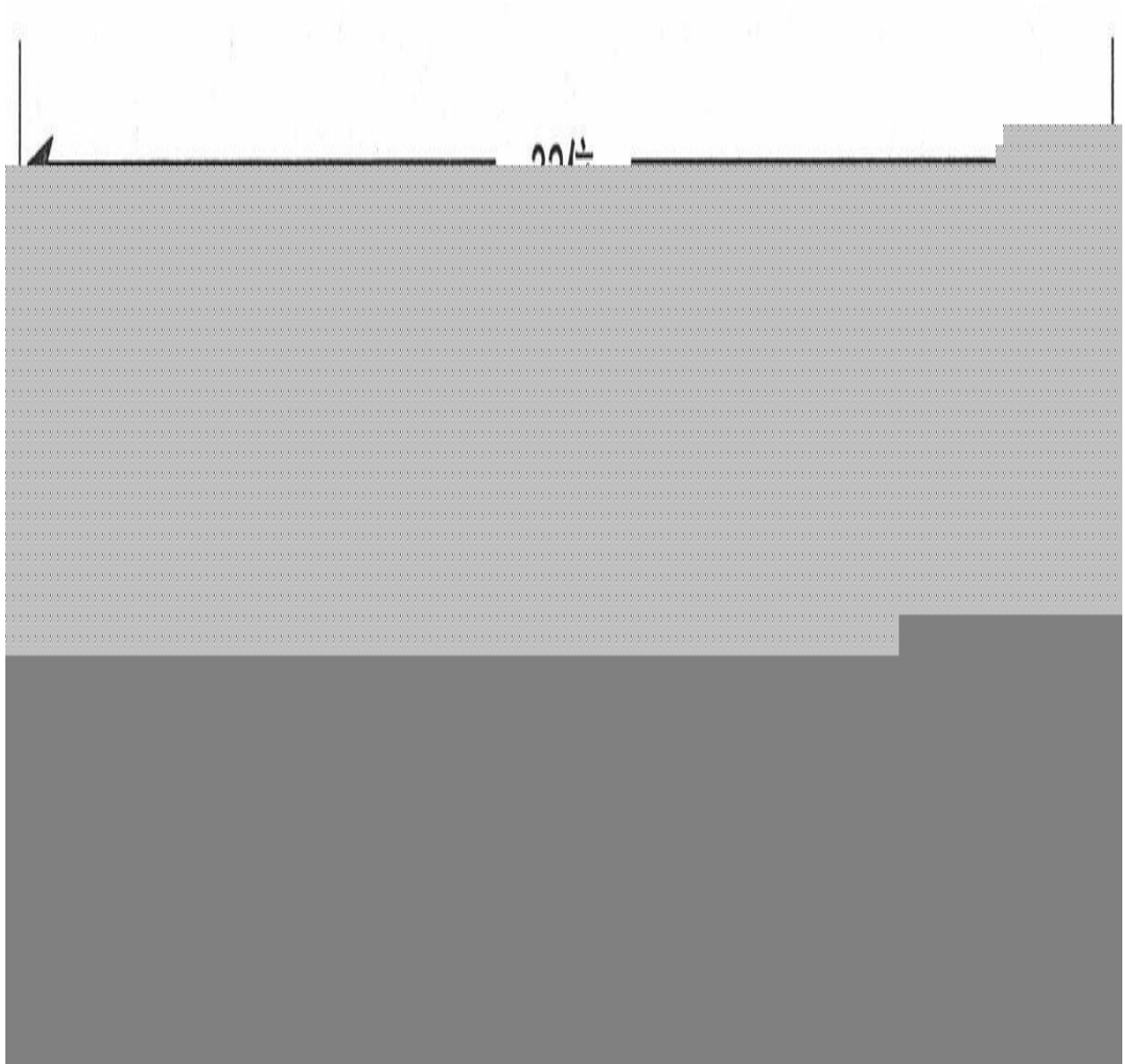
示例9-24 进一步使用**debug ipv6 ospf packet**调试命令显示出数据包封装失败



DBD数据包和Hello数据包不同，它不是多播传送的。DBD数据包是被发送到邻居路由器的IPv6地址的。回忆OSPFv3是使用链路本地地址进行包交换的，这可以通过IPv6数据包的调试命令输出看到。路由器Skrewt在接口Serial0/0上没有帧中继电路映射到地址FE80::202:FDFE:FE5A:E40上。这就是为什么会出现封装失败的原因。配置帧中继映射必须将邻居路由器的链路本地地址配置到本地DLCI上。

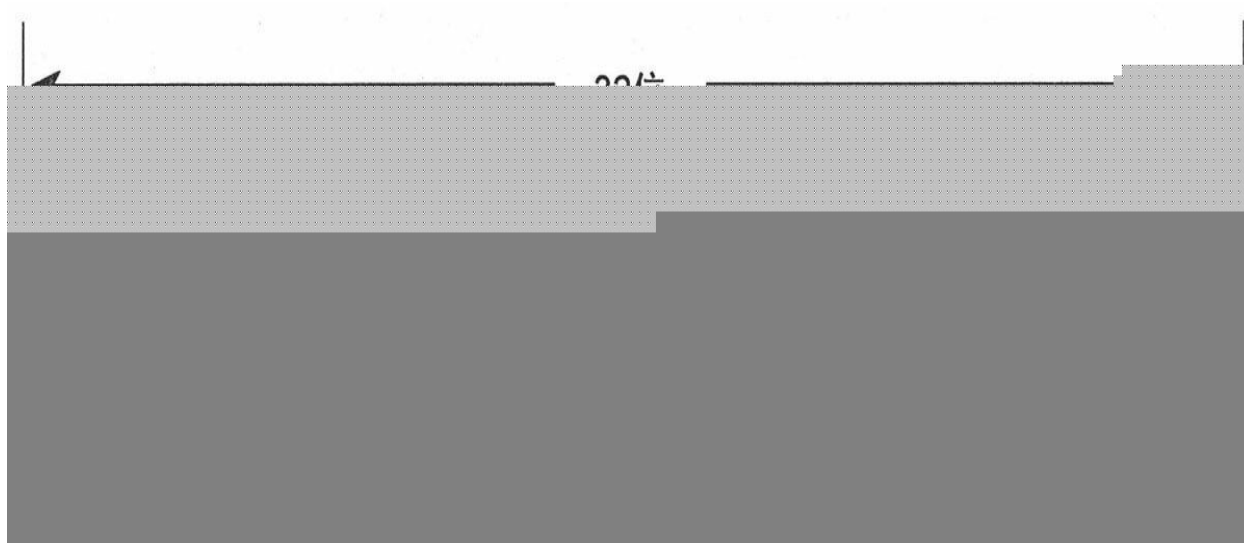
为了容易地获取IPv6接口的链路本地地址，可以使用命令**show ipv6 interface serial0/0**来查看，参见示例9-25所示。

示例9-25 使用命令**show ipv6 interface**可以看到有关接口的IPv6信息

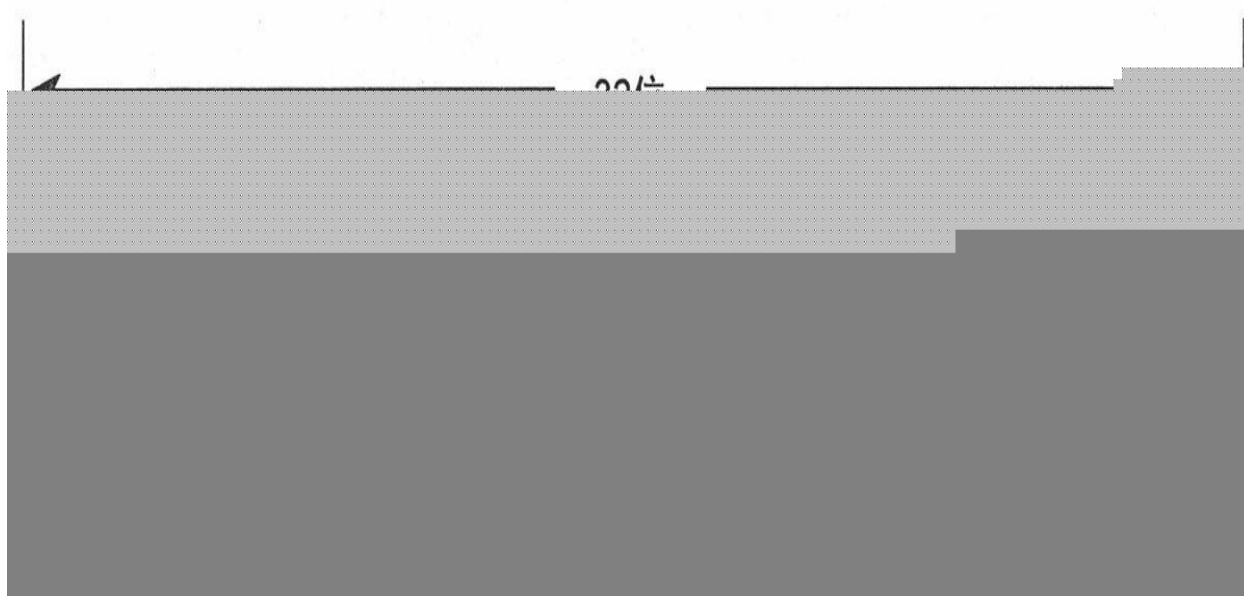


路由器Skrewt和Hippogriff更改后的配置显示在示例9-26和示例9-27中。

示例9-26 路由器**Skrewt**的帧中继映射配置更改为**IPv6**链路本地地址

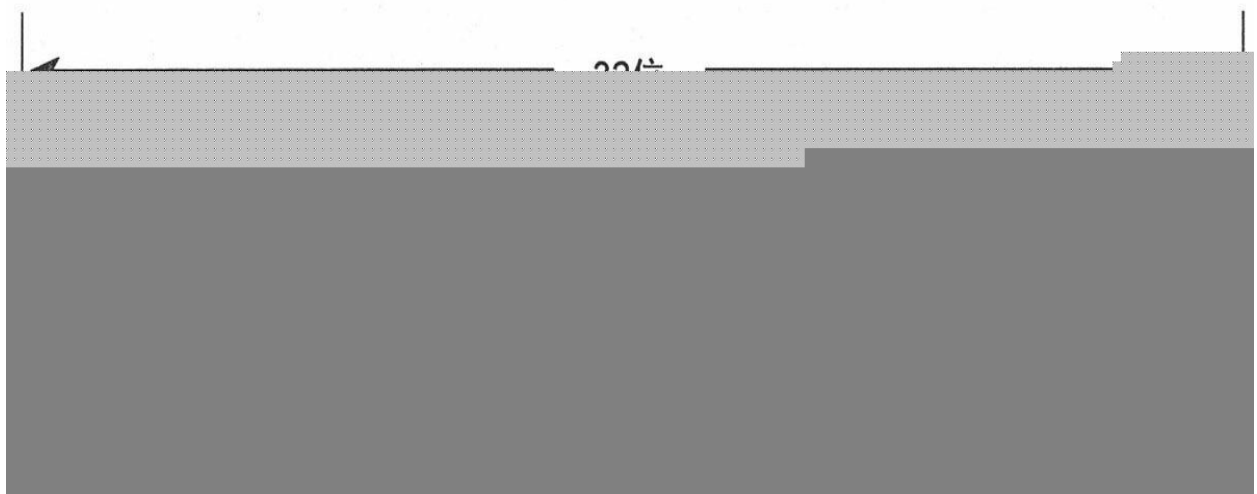


示例9-27 路由器**Hippogriff**的帧中继映射配置更改为**IPv6**链路本地地址



必须映射到DLCI上的IPv6地址是邻居的链路本地地址。参见示例9-28显示了路由器**Skrewt**正确的IPv6 OSPFv3邻居表。

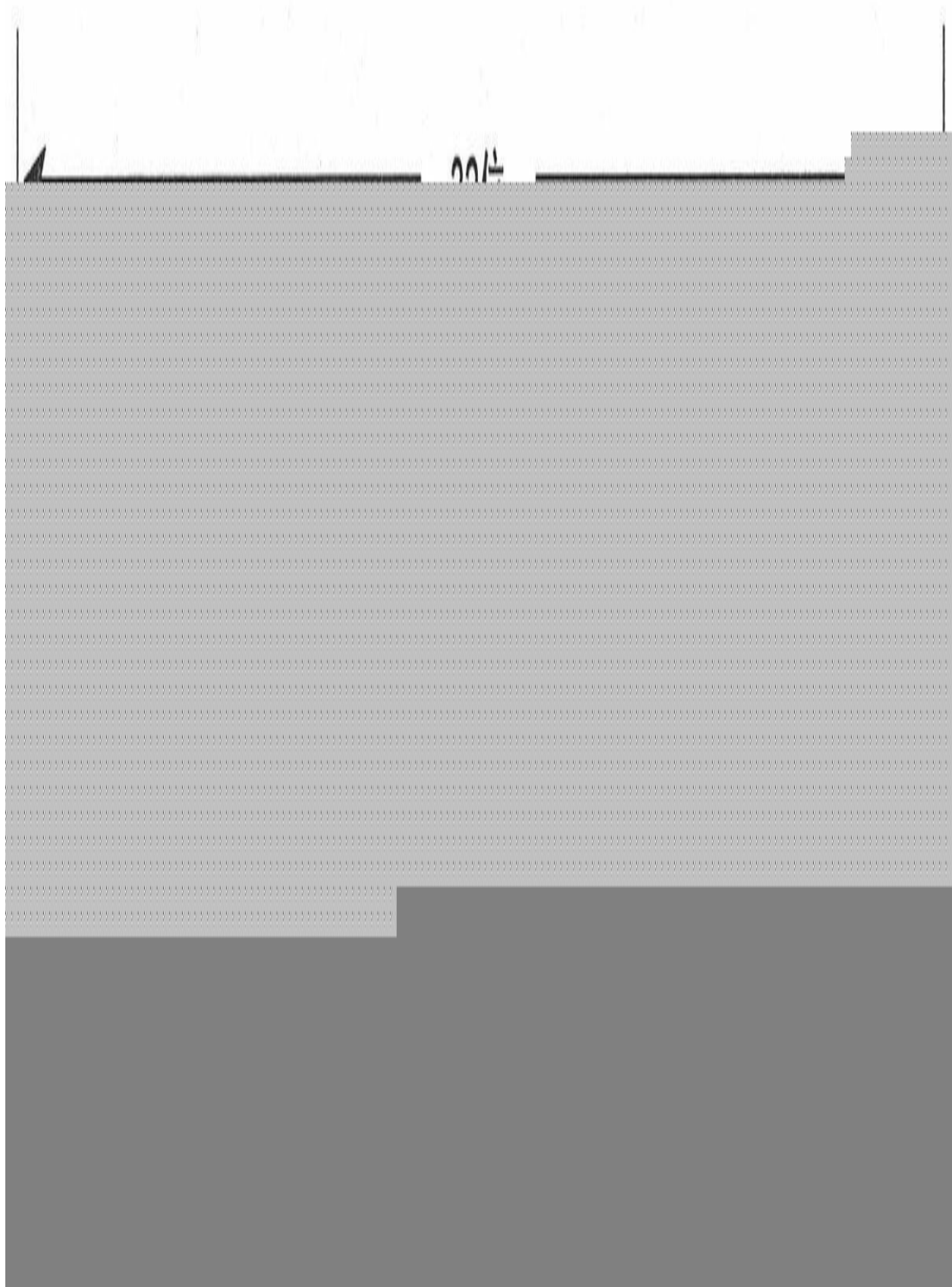
示例9-28 在将邻居的链路本地地址映射到**DLCI**后，路由器**Skrewt**与**DR**和**BDR**路由器建立了完整的邻接关系



9.4 展 望

我们用了两章的篇幅——其中一章篇幅还很长——描述了OSPF的主要内容。OSPF协议不仅是众所周知的链路状态路由选择协议，可能还是在IP路由选择协议中使用最为广泛的协议了。在下面的一章中，我们将讲述一个不太知名的链路状态协议：IS-IS协议。一些人认为它是一个“比较另类”的协议，但是实际上读者将会看到它相对于OSPF协议来说简单多了。

9.5 总结表：第9章命令总结



9.7 复习题

1. OSPFv3能够支持IPv4吗？
2. 在OSPFv3中，能够在每条链路上支持多个实例是什么意思？OSPFv3消息头部的什么字段可以实现这个目的？
3. OSPFv3数据包是怎么进行认证的？
4. OSPFv3的下一报头号是什么？
5. 两个保留的OSPFv3多播地址是什么？
6. OSPFv3是否使用了与OSPFv2不同的消息类型？
7. 在OSPFv3 LSA报头的链路状态类型字段中的起始3位的用途是什么？
8. 什么泛洪扩散范围是OSPFv3支持的，但OSPFv2不支持？这个泛洪扩散范围使用什么LSA？
9. 比较OSPFv3与OSPFv2中对应的路由器与网络LSA，它们最显著的不同之处是什么？
10. 区域内前缀LSA的用途是什么？
11. 链路LSA的用途是什么？

9.8 配置练习

1. 哪一个OSPFv3命令需要包含在OSPFv3进程的接口上已经配置的辅助IPv6前缀？
2. 如果一台路由器配置的IPv6地址为2001:db8:0:1::1/64，另一台路由器配置的IPv6地址为2001:db8:0:100::1/126，那么这两台路由器能够形成邻接关系吗？
3. 如图9-19所示，写出每一台路由器的IPv6 OSPFv3配置。
4. 在图9-19中，配置区域1成为一个完全末梢区域，并将地址汇总到区域0中。

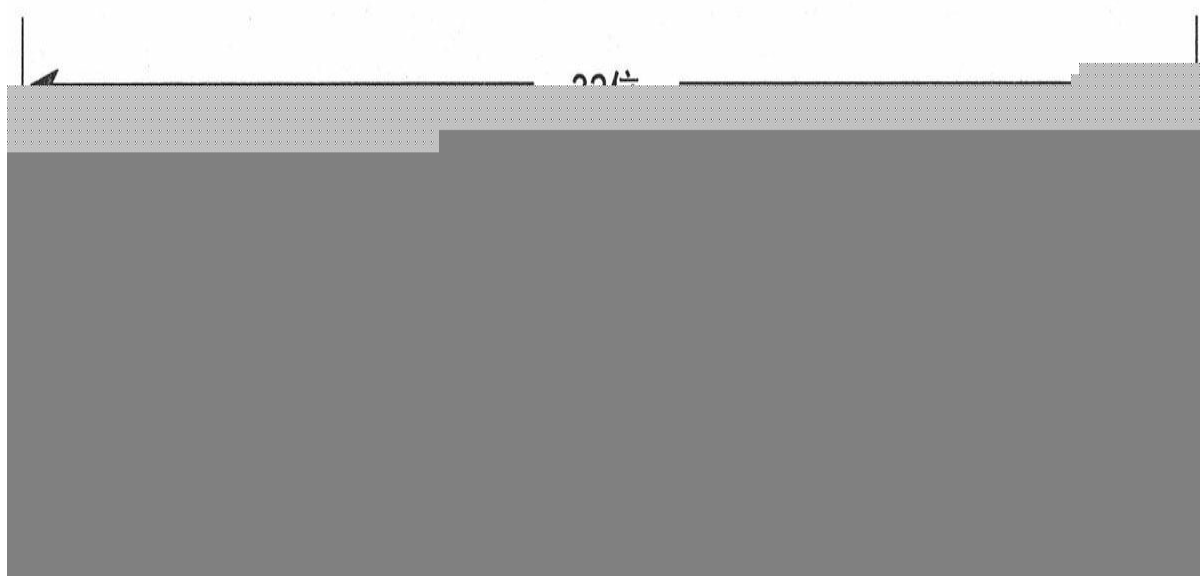


图9-19 IPv6 OSPFv3网络

[1] 在这里和其他OSPFv3 LSA中使用的接口ID不应该与IPv6地址的64位接口ID部分产生混淆。这个字段用的是32位的值，使这个接口区别于始发路由器上其他的接口。在RFC 2740中建议使用MIB—II If Index作为接口ID的值。

[2] IPv4使用帧中继反向ARP来动态地映射地址。在撰写本书时，IOS软

件还不支持动态地映射地址到帧中继PVC电路的IPv6功能。

[3] 不建议在一个实际运行的网络中调试IPv6数据包的信息。

本章包括以下主题：

- 集成IS-IS协议的基本原理与实现；
- 集成IS-IS协议的配置；
- 集成IS-IS协议的故障诊断。

第10章

集成IS-IS协议

每当人们提到链路状态协议和IP协议的术语时，大多数人会立即想到OSPF协议。一些人会说：“哦，是的，也有IS-IS协议，但是用得不多。”只有少数人会认真地考虑使用集成的IS-IS协议替代OSPF协议。但是，IS-IS协议的用户虽然为数不多但毕竟还是存在，在一些网络——主要是一些ISP和运营商网络——运行IS-IS协议进行IP路由选择。

IS-IS的意思是表示中间系统到中间系统，并且是为ISO无连接网络协议（ISO's Connectionless Network Protocol, CLNP）设计的路由选择协议。IS-IS协议是由ISO10589定义和解释的。[\[1\]](#)这个协议是由数字设备公司（DEC）的DECnet PhaseV第一次作为产品开发的。

ISO发展IS-IS协议的时间和IAB（Internet Architecture Board, Internet体系结构委员会）发展OSPF协议的时间基本是同一时期，只是稍早或稍迟一点而已。并且提议采用IS-IS协议替代OSPF协议作为TCP/IP协议的路由选择协议。驱动该提议的观点是，即TCP/IP协议只是一个过渡的协议簇，并且最终会被OSI协议簇代替。向着OSI发展的推动力来自一些技术规范，例如GOSIP和EPHOS等。GOSIP（United States' Government Open Systems Interconnection Profile）是指美国政府开放系统互连框架文件，EPHOS（European Procurement Handbook for Open Systems）是指欧洲国家关于开放系统的采购手册。

为了支持从TCP/IP协议向OSI协议可预见的转换，又提出了一个扩展的IS-IS协议，[\[2\]](#)称为集成IS-IS协议。提出集成IS-IS协议的目的是为了把它作为一个具有双重功能的IS-IS协议，即利用单个路由选择协议同时为CLNS协议[\[3\]](#)和IP协议提供路由选择的能力。这个协议可以设计用来在一个单纯的CLNS环境，一个单纯的IP环境，或者一个CLNS/IP的混和环境运行。

说是画了一条战线可能有点过分夸张，但是至少是形成了两个鲜明的派别——ISO的支持者和OSPF的支持者。读者应该阅读和对比一些经典的书籍，其中关于OSPF和IS-IS的讲述是很有启发作用的。这些书籍是由

Christian Huitema^[4]——IAB的前任主席和Radia Perlman^[5]——IS-IS的首席设计师编写的。最后，IETF（Internet工程任务组）采用了OSPF协议作为建议使用的IGP协议。技术上的不同的确会影响到决心，但是，有时也会存在行政上的因素。ISO标准化是一个缓慢的处理过程，它一般需要4个步骤，并且依赖多个委员会最终投票表决同意。而IETF的处理过程却快捷得多。可以看出，通过RFC进程，发展OSPF协议要比采纳拘泥于形式化的IS-IS协议更有意义。

另一方面，非常小但却非常复杂的IS-IS用户群体证明IS-IS协议对于它的实施者与用户是有好处的。支持该协议的扩展特性很快就达成一致，因为IS-IS的用户大多数是高端的ISP和运营商，他们对他们的路由器厂商具有很高的要求。对于任何一个希望进入高性能网络市场的路由器厂商来说，它们都必须升级自己的产品以支持IS-IS协议。

不考虑行政上的争议，OSPF工作组实际上学习和利用了很多IS-IS设计中的基本机制。从表面看来，OSPF协议和IS-IS协议有很多共同的特性：

- 它们都维护一个链路状态数据库，并且这个数据库都是来自一个基于Dijkstra的SPF算法计算的一棵最短路径树；
- 它们都利用Hello数据包来形成和维护邻接关系；
- 它们都使用区域的概念来构成一个两级层次化的拓扑结构；
- 它们都具有在区域之间提供地址汇总的能力；
- 它们都是无类别路由选择协议；
- 它们都通过选取一个指定路由器来描述广播型网络；
- 它们都具有认证的能力。

除了这些类似之处外，它们也有明显的不同。本章将通过检查它们的这些不同之处开始讲述。本章只把集成IS-IS协议（以下简称为IS-IS协议）作为一个IP路由选择协议来讲述，只有在使用IS-IS协议为IP协议路由以及与CLNS协议有关的地方时才讲述CLNS协议。正如前面已经提到的，IS-IS协议几乎是专门用在服务提供商网络中的，这样的网络往往对

可靠性与可扩展性具体很高的要求；因此，本章另一个重点就是研究IS-IS协议在特大型网络中满足某些特殊要求的特性。

10.1 集成IS-IS协议的基本原理与实现

ISO经常使用不同的术语来描述IETF所描述的相同概念实体，这种情况有时会引起混淆。ISO的术语将在本节介绍和定义，但是在一般情况下，本章将使用本书其余章节使用的更类似于IETF的术语。[\[6\]](#)有一些ISO的术语是非常基本的，因此，在具体讲述IS-IS协议的所有术语之前先介绍一下这些术语。

一台路由器就是一个中间系统（Intermediate System, IS），而一台主机就是一台端系统（End System, ES）。因此，提供主机与路由器之间通信的协议称为ES-IS协议，而被路由器用来进行相互宣告的协议（路由选择协议）称为IS-IS协议（如图10-1所示）。虽然IP协议使用路由器发现机制，例如Proxy ARP、IRDP，或IPv6 NDP，或者在主机上配置简单的缺省网关，但是CLNP协议还使用ES-IS协议来形成端系统和中间系统之间的邻接关系。对于IP协议来说，ES-IS协议和IS-IS协议没有什么相关之处，因此本书将不包括有关ES-IS协议的讨论。

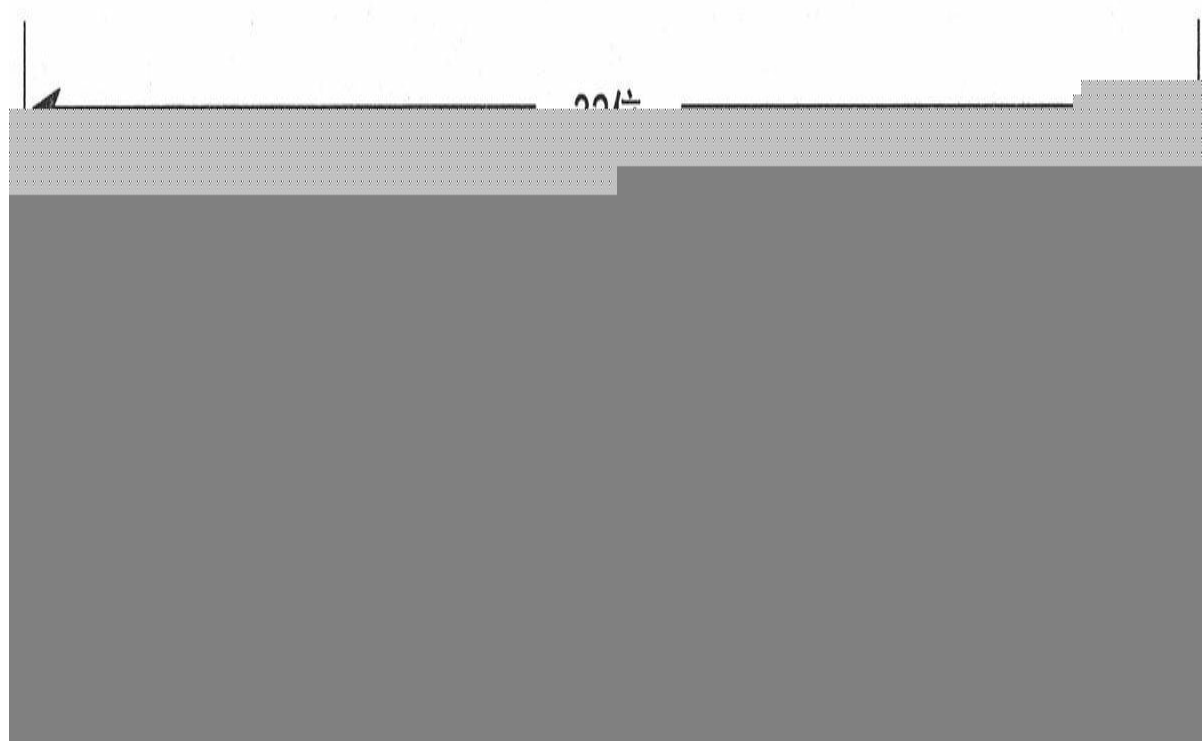


图10-1 在ISO的术语中，主机是端系统，而路由器是中间系统

与一个子网相连的接口称为子网连接点（Subnetwork Point of Attachment, SNPA）。SNPA有一些概念化，因为它实际上是定义了一个提供子网服务的“点”，而不是一个实际的物理接口。SNPA的基本概念特性和子网本身的基本概念特性是相符合的，它可以由数据链路交换机相连的多个数据链路组成。

从一个节点的OSI层到另一个节点对等的OSI层的数据单元称为协议数据单元（Protocol Data Unit, PDU）。因此，一个帧就是一个数据链路PDU（DLPDU），而一个数据包（或者分组）就是一个网络层协议数据单元（NPDU）。执行与OSPF协议中的LSA等价功能的数据单元称为链路状态PDU（LSP）[\[7\]](#)。但与LSA不同，LSA是封装在OSPF头部之后的，并且都被封装在一个IP数据包内，而一个LSP本身就是一个数据包。

10.1.1 IS-IS区域

虽然IS-IS协议和OSPF协议都使用区域的概念来创建两级的层次化网络拓扑结构，但是它们存在一个基本的不同之处，就是这两种协议在定义区域的方法上不一样。如图10-2所示，OSPF协议的区域边界是通过路由器来划分的。某些接口属于一个区域，而另一些接口属于其他区域。如果一台OSPF路由器具有的接口分布在多于一个的区域里，那么这台路由器就是一台区域边界路由器，即ABR路由器。

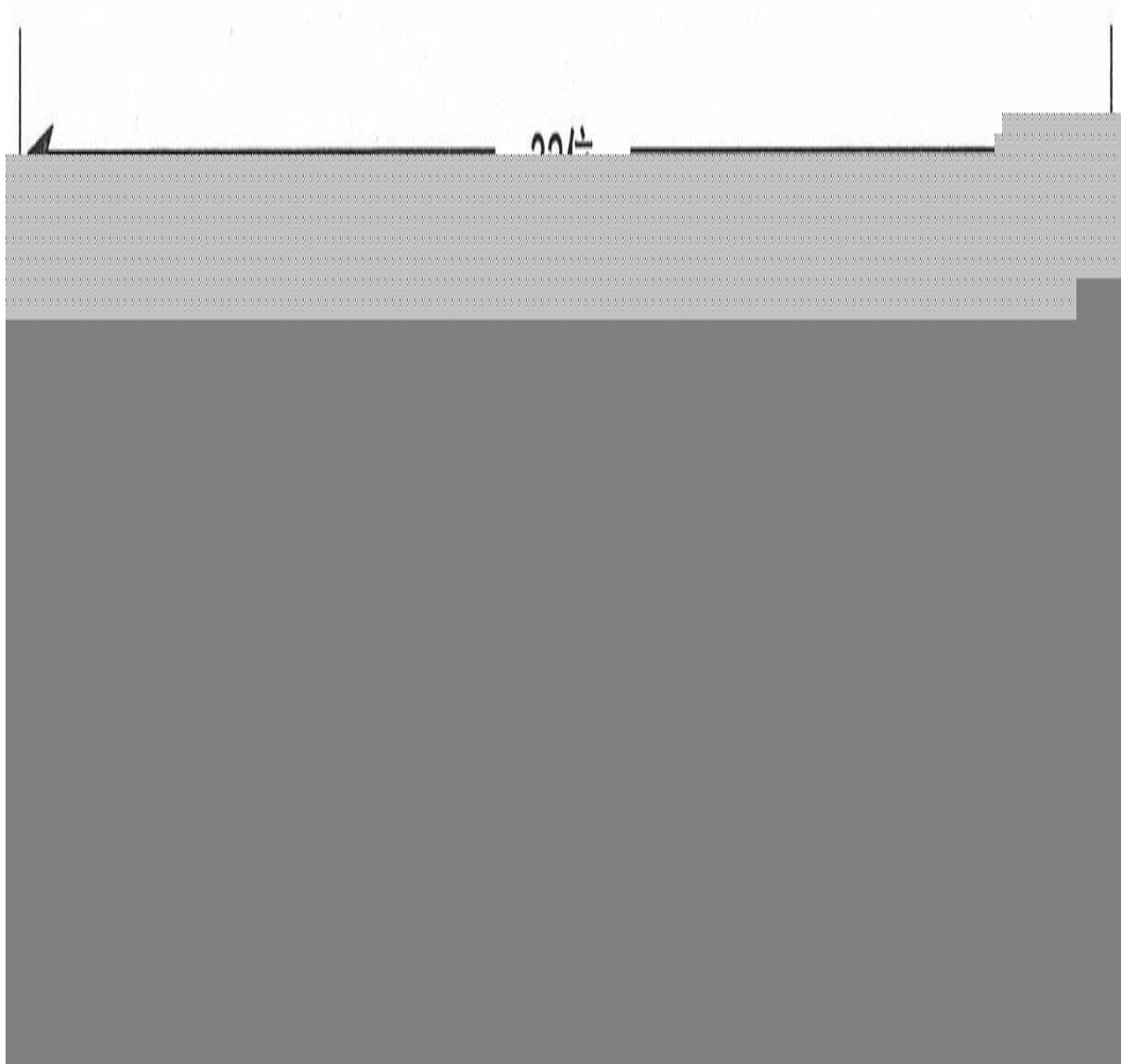


图10-2 OSPF区域边界是在路由器上，而与不同区域相连的路由器是ABR路由器

图10-3显示了与图10-2完全相同的网络拓扑，只是把它设计成了IS-IS区域。注意，所有的路由器都完全处在一个区域内部，并且区域的边界是在链路上，而不是在路由器上。IS-IS“骨干区域”是第2层区域，而非骨干区域是第1层区域。

一个中间系统可以是一台第1层的路由器（L1）、一台第2层的路由器（L2）或者两种类型皆是的路由器（L1/L2）。L1路由器类似于OSPF协议中的非骨干内部路由器，而L2路由器类似于OSPF协议中的骨干路由

器，同样地，L1/L2路由器类似于OSPF协议中的ABR路由器。在图10-3中，L1/L2路由器和L1路由器以及L2路由器相连。这些L1/L2路由器必须同时维护一个L1的链路状态数据库和一个L2的链路状态数据库，这种方式和OSPF协议中ABR路由器必须维护与之相连的每一个区域各自的数据库相类似。在Cisco 路由器上，可以使用命令**is-type** 来配置L1-only、L2-only或L1/L2类型，缺省情况下配置为L1/L2。

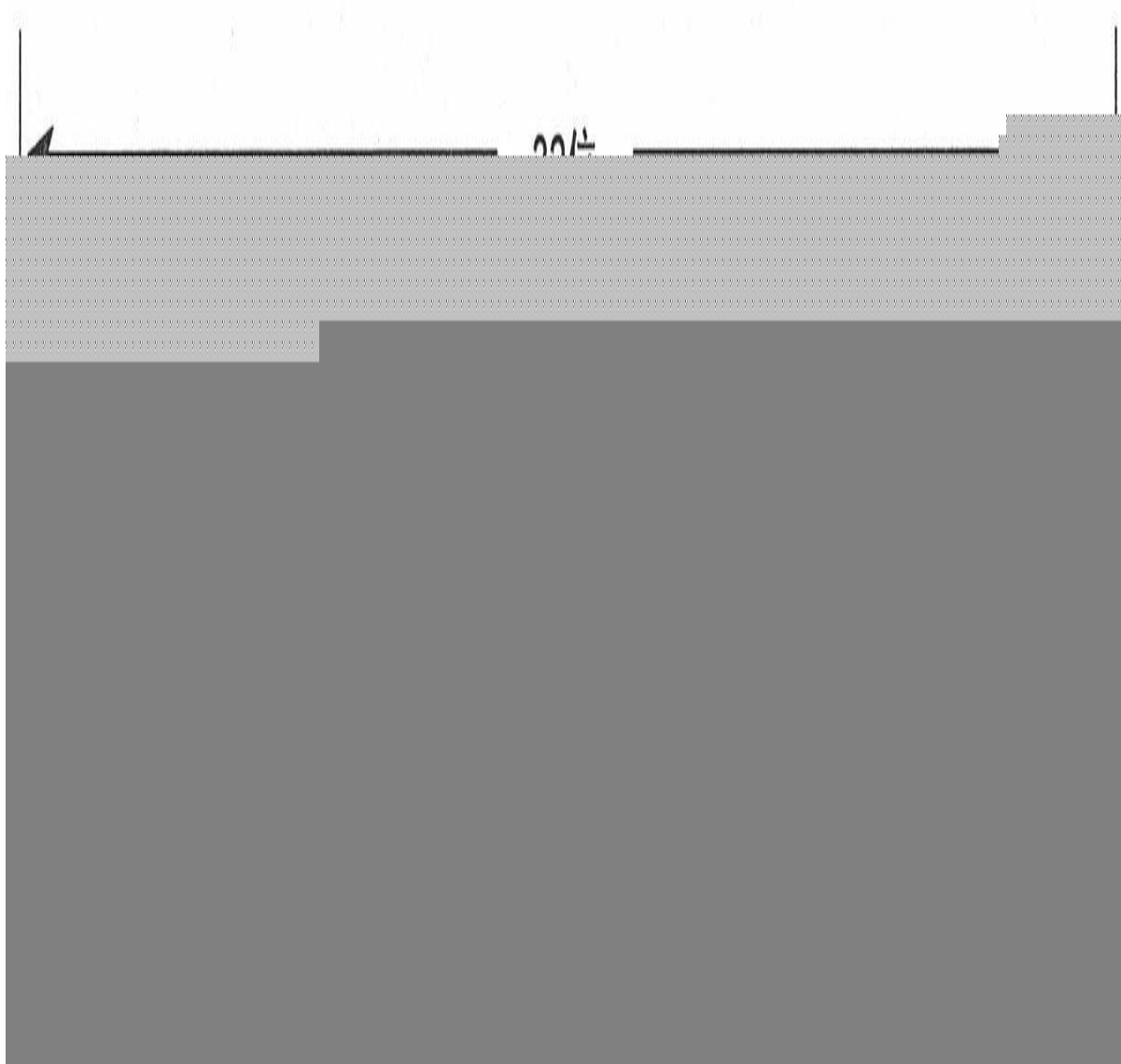


图10-3 IS-IS区域的边界在链路上，而与区域相连的路由器是第2层路由器

图10-3中的区域拓扑图对于比较IS-IS区域和OSPF区域是有帮助的，它

也会带来一些误解；IS-IS区域不像OSPF区域那样显得清楚明了。与其把IS-IS区域当作拓扑上的区域，倒不如把它们理解成一组邻接关系更好一些。一个邻接可以是L1邻接也可以是L2邻接。例如，一个特定的L1区域实际上是具有相同AID的路由器的一组相邻的L1邻接。L2区域总是定义为一组L2邻接，理解这一点非常重要。一台L1/L2路由器是位于L1区域内的路由器，它可以同时具有一条或多条L1邻接和一条或多条L2邻接。一台L2路由器仅仅具有L2邻接关系。

在两个邻居之间同时存在一条L1邻接和一条L2邻接也是可能的。因为IS-IS区域被定义为一系列的邻接，在相同的两个邻居之间同时具有L1邻接和L2邻接意味着IS-IS区域是可以重叠的。与OSPF协议相比，IS-IS的区域边界不像OSPF那么清楚。

与OSPF协议相同，区域间的通信量都必须经过L2区域，以便防止区域间路由选择环路。在一个区域内的每台L1路由器（包括区域内的L1/L2路由器）都会维护一个同样的链路状态数据库。但与OSPF协议中的ABR路由器不同，缺省情况下，L1/L2路由器不需要通告L2类型的路由给L1类型的路由器。因此，一台L1路由器无法知晓它自己所在区域之外的目的路由。在这个意义上，一个L1区域就相当于OSPF协议中的完全末梢区域。为了路由转发数据包到其他的区域，L1路由器必须转发数据包到一台L1/L2路由器上。当L1/L2路由器发送它的第1层LSP进入一个区域时，它将通过在LSP中设置一个称为“区域关联位（Attached, ATT）”[\[8\]](#)的二进制位来通知其他L1路由器它可以到达其他的区域。

ISO 10589描述了IS-IS协议路由器可以利用虚链路来修复被分段的区域，这和OSPF是一样的。但是这个特性在Cisco路由器和其他大多数厂商的路由器上都不支持，因而不在这里作进一步的描述。厂商不支持IS-IS虚链路的主要原因很简单，就是他们的客户不要求他们支持，这是IS-IS与OSPF在应用中的一个基本不同之处。OSPF协议可以设置丰富的区域工具和特性，是企业网络选用的协议。另一方面，多区域的IS-IS运行更为复杂，因此很少在企业网中出现。运营商和ISP运行的是基于BGP协议的大型网络，他们的IGP协议主要用来寻找BGP会话的节点。因此，他们希望他们的IGP协议尽可能的简单——通常会把他们的整个路由选择域作为单个IGP区域。IS-IS协议在很多方面无疑比OSPF协议更为简单，这对于“单个大型区域”这种类型的应用来说更具有可扩展性。因此，当读者在一个服务提供商的网络中碰到IS-IS协议，你通常会遇到单一的L2区域。[\[9\]](#)

由于一台IS-IS路由器可以完全地处于一个单一的区域，因此区域ID（或区域地址）将和整个路由器相关联，而不是和某一个接口相关联。IS-IS协议的一个独特特性是，在缺省情况下一台路由器最多可以拥有3个区域地址，这在区域过渡期间是很有用的。利用IOS软件的命令**max-area-addresses**，读者能够将区域地址的数目增加到最多254个。在10.2.3小节中演示了多个区域地址的使用。每台IS-IS路由器必须拥有一种在它所在的路由选择域内唯一地标识它本身的方法。这个唯一标识就是系统ID的功能，系统ID（System ID）类似于OSPF协议中的路由器ID。在一台IS-IS路由器上可以通过一个单一的地址同时定义区域ID和系统ID，这个地址就是网络实体标题（Network Entity Title, NET）。

10.1.2 网络实体标题

即使IS-IS协议只用来为TCP/IP协议进行路由选择，它也依然是一个ISO CLNP协议。因此，IS-IS协议对等体之间的通信数据包是CLNS PDU；也就是说即使是在一个纯IP环境中，一台IS-IS路由器也必须有一个ISO地址。这个ISO地址是一个网络地址，称为网络实体标题（NET），并在ISO 8348中进行描述。[\[10\]](#)一个NET地址的长度范围可以是8~20个八位组字节，并可以描述为区域ID和一台设备的系统ID两部分，如图10-4所示。

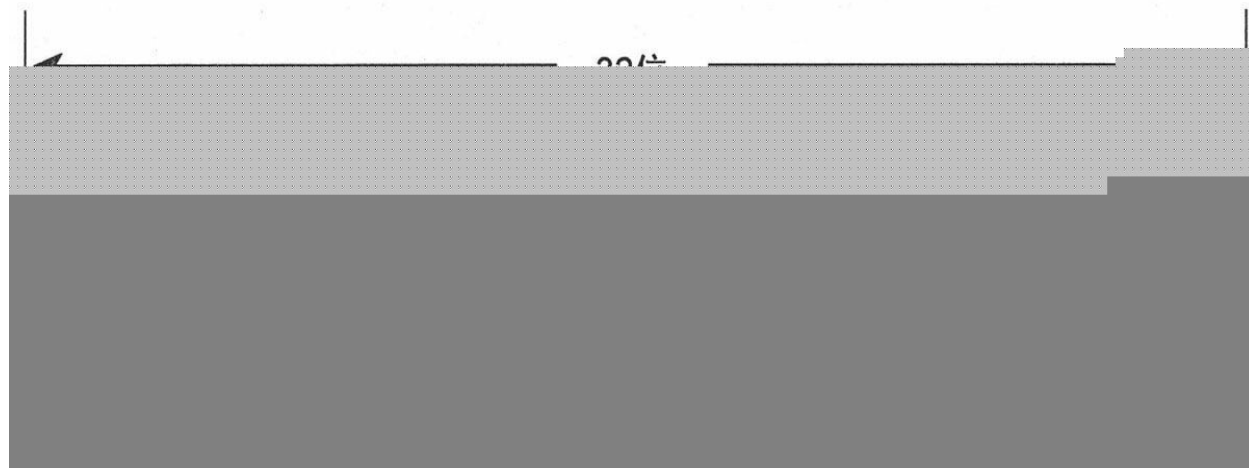


图10-4 NET地址指定了区域ID和一个IS或者ES的系统ID

ISO设计的NET地址可以在许多系统中做很多事情，这依赖于你个人的看法，要么认为这个地址的格式是非常灵活的和可扩展的，要么认为这个地址格式是一个麻烦的容易搞糊涂的变量字段。如图10-5所示，图中

仅仅显示了一个ISO NET地址可能具有的多种格式中的3种。虽然在每一个例子中系统ID前面的域是不同的，但是系统ID本身都是相同的。ISO 10589指定了这个域的长度可以从1~8个八位组字节，但是在一个路由选择域内的所有节点的系统ID必须使用相同的长度。实际上，这个系统ID的长度是6个八位组字节，[\[11\]](#)并且经常是这台设备上的某个接口的MAC地址（Media Access Control，介质访问控制）。对于路由选择域内的每一个节点，这个系统ID必须是惟一的。

在图10-5的例子中还有一个需要注意的地方，就是NSAP选择符（SEL）。在所有的情况下，这个1个八位组字节的字段都被设置为0x00。一个网络服务接入点（NSAP）所描述的都和某个节点在网络层上的一种特有服务相关联。因而，在一个ISO地址中，SEL设置为大于0x00的某些值时，这个地址就是一台NSAP地址。这种情况和一个IP数据包内的IP目的地址与IP协议号的组合有些类似，它表明一台具体设备的TCP/IP协议栈的网络层上的一个具体服务。而在一个ISO地址的SEL设置为0x00时，这个地址就是一个NET地址，指明了某个节点网络层本身的地址。

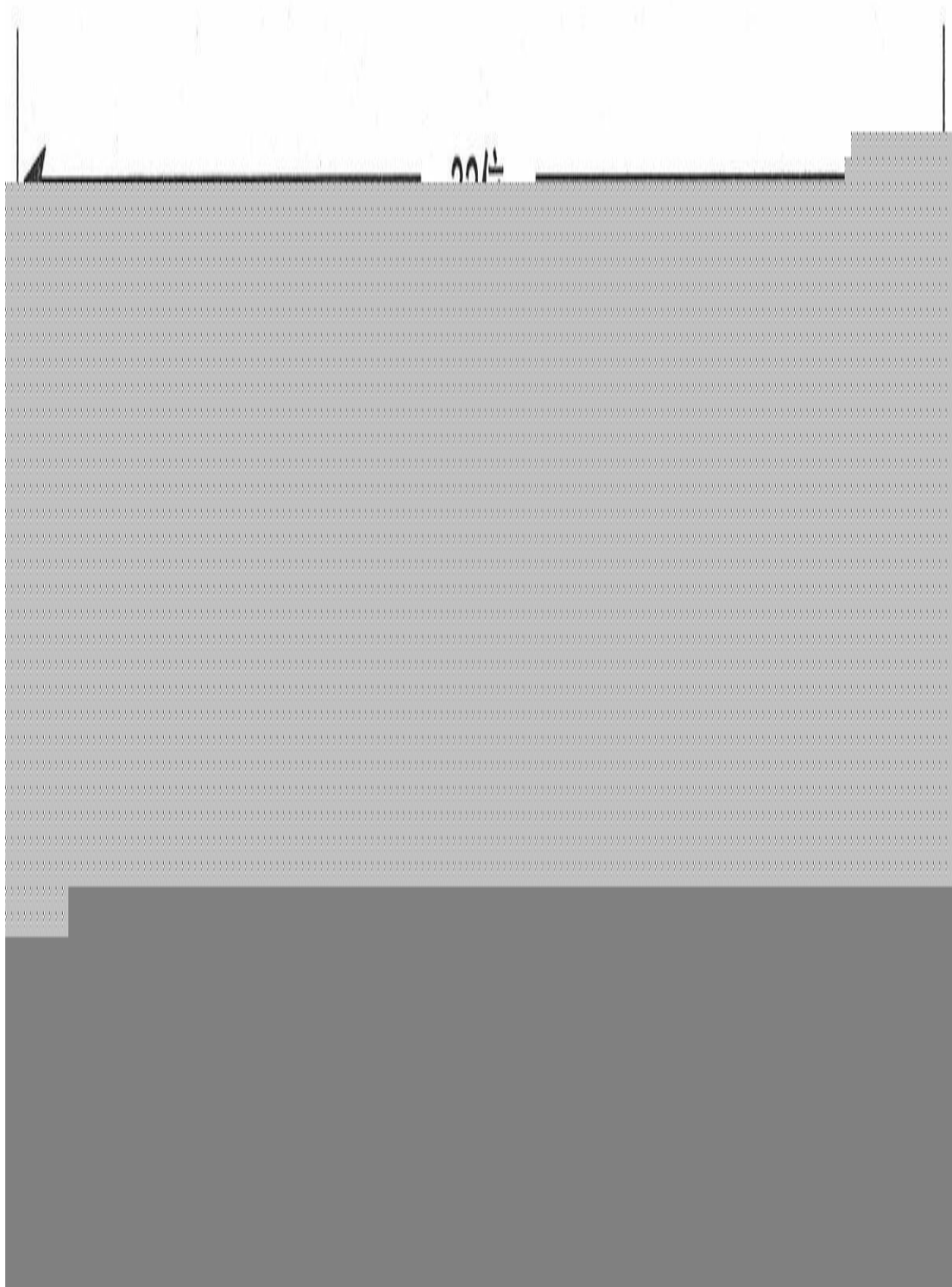


图10-5 3 种NET格式：一个是简单的8个八位组字节区域ID/系统ID格式（a）；一个是OSI NSAP格式（b）；还有一个是GOSIP NSAP格式 [12] （c）

图10-5中只是显示了NET的多种格式，关于NET配置更详细的讨论已经超出了本书的讨论范围。如需要进一步的学习，RFC 1237是一个不错的参考。 [13] 在只有IP协议的环境中，NET地址的设定可以基于某个标准，例如GOSIP。如果你可以在一个纯IP环境中自由地选择任何NET地址的格式，那么将可以根据实际网络的需要选择最简单的格式。

无论是何种格式的地址，都需要满足下面的3个规则：

- NET地址必须以一个单个八位组字节的域开始（例如，47.xxxx...）；
- NET地址必须以一个单个八位组字节的域结束，并且应该设置为0x00 (...xxxx.00)。如果SEL是非零的，IS-IS也会起作用，但是在一个CLNP/IP混和的路由器可能会出现一些问题；
- 在Cisco的路由器上，NET地址的系统ID必须是6个八位组字节。

10.1.3 IS-IS的功能结构

像ISO模型一样，之所以有一个分层的网络体系结构，其中有一个最主要的目的是为了使每一层的功能都独立于它下面的一层。例如，网络层必须适应大多数类型的数据链路或子网络。为了进一步满足这种适应性，网络层又由两个子层组成（如图10-6所示）。子网独立子层

（subnetwork independent sublayer）为传输层提供了一致的和统一的网络服务；而子网依赖子层（subnetwork dependent sublayer）则为子网独立子层的需求而去存取数据链路层提供的服务。正如这两个命名所暗示的，子网依赖子层取决于数据链路的具体类型，而子网独立子层则能够独立于数据链路。

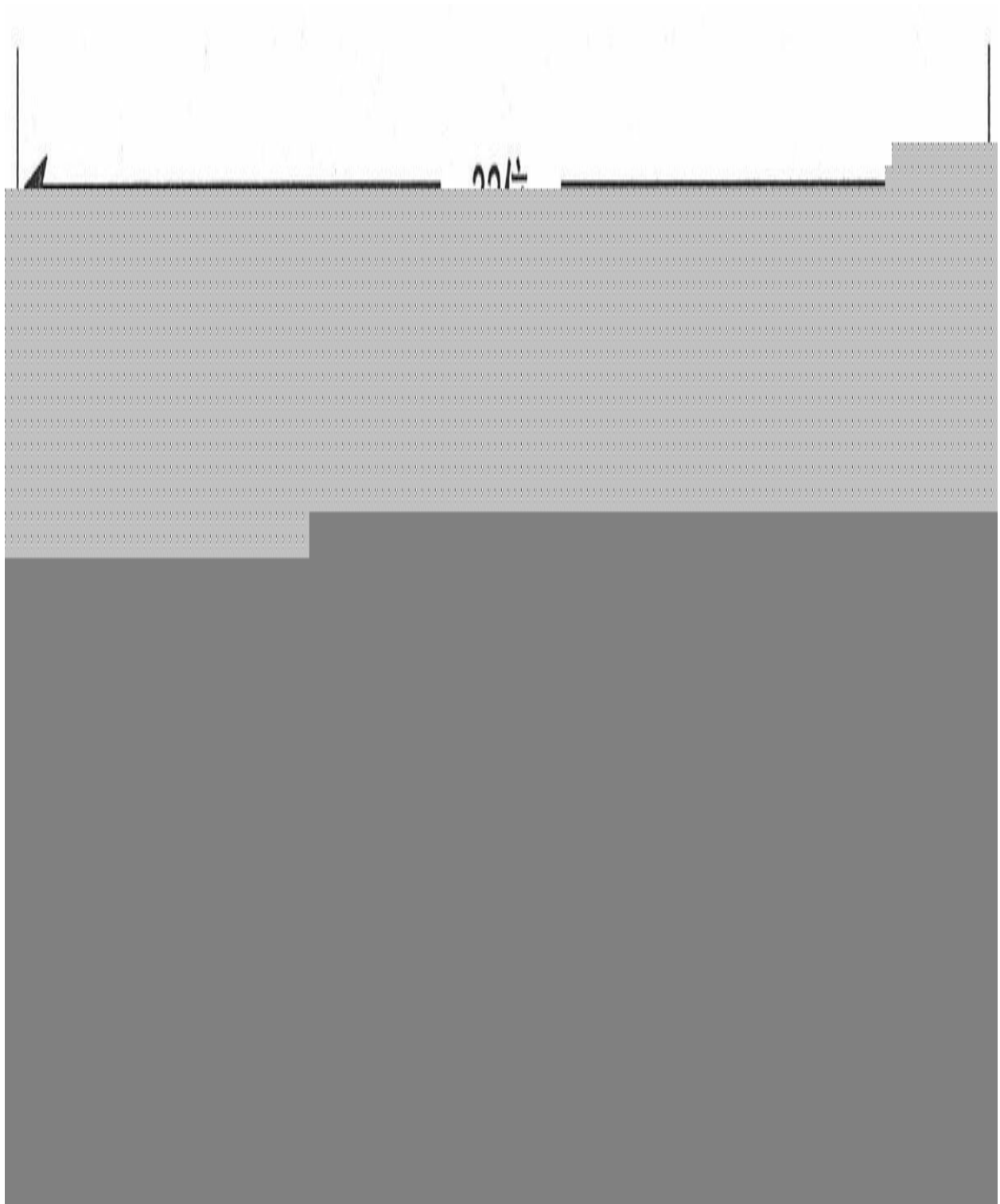


图10-6 OSI网络层由两个子层组成

网络层的结构是在ISO 8648 [\[14\]](#)中指定的，它实际上要比图10-6中显示

的结构更为复杂。这里之所以提到这两个基本的子层，是因为在ISO 10589中对这些子层的功能架构中有关IS-IS的操作做了大量描述。

1. 依赖于子网的功能

依赖于子网的功能是指子网的功能依赖于它的下层；这是指为子网独立子层的功能层面隐藏掉不同种类数据链路（子网）的特征。下面的依赖于子网的功能对于路由选择是非常重要的：

- PDU数据包的传送和接收是在一个具体相连的子网上；
- 通过IS-IS的Hello PDU数据包来发现邻居路由器并建立这个子网上的邻接关系；
- 邻接关系的维护；
- 链路的复用，或者说对于OSI进程转换为OSI PDU数据包，而对于IP进程转换为IP数据包。

（1）IS-IS网络类型

相对于OSPF协议中定义的4种网络类型，IS-IS协议只定义了两种类型：广播型子网和 点到点或一般拓扑子网。广播型子网的定义是和OSPF协议中的定义一样——就是支持多路广播的多路访问数据链路。而点到点子网（非广播子网）则可能是永久链路，例如T1链路，或者动态链接链路，例如X.25的SVC链路。

（2）邻居路由器和邻接关系

IS-IS路由器是通过交换IS-IS Hello PDU数据包信息来发现邻居并形成邻接关系的。Hello数据包每隔10s传送一次，在Cisco的路由器中，这个时间间隔可以基于每个接口通过命令**isis hello-interval** 来改变。虽然IS-IS Hello数据包对于广播型子网和点到点子网略有差别，但是Hello数据包中包含的基本信息还是相同的，这将在10.1.4小节中讲述。一台IS-IS路由器使用它的Hello PDU数据包可以标识它本身和它的性能，以及描述发送该Hello数据包的接口的一些参数。如果两台邻居路由器关于它们各自的性能和接口的参数协商一致，那么它们就形成了邻接关系。然而，IS-IS在形成邻接关系方面没有OSPF那样严格。在大多数实例中，

一个邻居通告的特性如果其他邻居不支持的话不会阻止它们形成邻接关系；其中的特性可以被忽略。邻居之间甚至可以通告不同的Hello时间间隔。

对于第1层类型与第2层类型的邻居路由器，IS-IS协议可以形成各自不同的邻接关系。L1-only路由器可以和L1以及L1/L2邻居形成L1邻接关系，而L2-only路由器可以和L2以及L1/L2邻居形成L2邻接关系。L1/L2路由器和它的邻居既可能形成L1邻接关系也可能形成L2邻接关系。但是，一台L1-only路由器和一台L2-only路由器不能形成一个邻接关系。正如前面所谈到的，缺省情况下，Cisco路由器是L1/L2路由器。

当路由器的类型（L1-only、L2-only或L1/L2）影响所形成的邻接类型时，在两台邻居路由器上配置的区域ID也会对其产生影响。这可以应用以下的规则：

- 两台L1-only路由器只有在它们的AID匹配时才能形成一个L1邻接关系；
- 两台L2-only路由器即使它们的AID不同也能够形成一个L2邻接关系；
- 一台L1-only路由器和一台L1/L2路由器只有在它们的AID匹配时才能形成一个L1邻接关系；
- 一台L2-only路由器和一台L1/L2路由器即使在它们的AID不同时也能形成一个L2邻接关系；
- 如果两台L1/L2路由器的AID匹配，它们就可以同时形成L1和L2类型的邻接关系；
- 如果两台L1/L2路由器的AID不匹配，它们就只能形成L2类型的邻接关系。

一旦邻接关系建立成功，Hello数据包将担当保活（keepalive）的功能。每一台路由器都在它的Hello数据包中发送一个抑制时间（hold time），用来通知它的邻居路由器在宣告这台路由器无效之前，应该等待多长的时间去侦听下一个Hello数据包。在Cisco路由器上，缺省的抑制时间是Hello时间间隔的3倍长，并且可以基于每接口通过命令**isis hello-**

multiplier 来改变。与OSPF的一个重要不同之处是，两个IS-IS邻居之间的Hello时间间隔与保持时间是不需要匹配的。每一台路由器都会认同它的邻居通告的保持时间。

OSPF与IS-IS邻接的另一个有趣的不同之处是，两台路由器形成邻接关系的时候。OSPF协议在这个地方有点混乱——两台路由器一旦建立了双向通信就认为形成了邻接关系，但直到数据库完成了同步才认为形成了完全邻接关系。而IS-IS协议在路由器之间能够交换Hello数据包时就认为它们形成了邻接关系。

如示例10-1所示，IS-IS的邻居表可以通过命令**show clns is-neighbors** 来查看。示例中显示的开始4列表明了每一台邻居路由器的系统ID、与邻居相连的本地接口、邻接关系的状态以及邻接关系的类型。这里的状态要么是Init——表明邻居路由器是学习到了但是还没有形成邻接关系，要么是Up——表明和邻居路由器成功建立了邻接关系。优先级是指在广播型网络上用来选举指定路由器的路由器优先级，这将在下面介绍。

示例10-1 IS-IS的邻居表可以通过命令**show clns is-neighbors**来显示

22位

路由器的系统ID是0000.0c76.5b7c，而伪节点ID是02。

最后一列指出了邻接关系的格式。对于集成的IS-IS协议，这个格式将永远是Phase V，用来说明是OSI/DECnet Phase V。另外一个惟一的格式是DECnet Phase IV。

（3）指定路由器

IS-IS协议在一个广播型多路访问网络上选取指定路由器（更为准确地说，是一个指定IS）的原因与OSPF协议相同。把网络本身看作是一台路由器或一个伪节点，要比局域网内的每一台路由器都要与该网络上相连的其余每台路由器形成一个邻接关系的方法好得多。包括指定路由器在内的每一台路由器都只需要通告单条链路到伪节点。指定路由器作为伪节点的代表也会通告一条链路到与之相连的所有路由器。

然而，与OSPF协议不同的是，与广播型多路访问网络相连的IS-IS路由器要与网络上的所有邻居建立邻接关系，而不仅仅是指定路由器。这和前面章节所讲述的是一致的，一旦可以交换Hello数据包就建立了IS-IS邻接关系。每一台路由器将以组播方式发送它的LSP数据包给所有的邻居路由器，并且指定路由器使用一个称为序列号PDU（Sequence Number PDU, SNP）的数据包来确保LSP的泛洪是可靠的。这个可靠的泛洪过程和SNP数据包将在后面的“更新过程”部分中介绍。

IS-IS协议指定路由器的选取过程非常简单。每一台IS-IS路由器接口都被指定一个L1类型的优先级和L2类型的优先级，它们的范围是0～127。Cisco路由器接口的优先级对于L1和L2类型的缺省值都是64，并且可以通过命令isis priority来改变这个值。

路由器通过其每一个接口发送Hello数据包，并在Hello数据包中通告它的优先级——在L1类型的Hello数据包中通告L1类型的优先级，在L2类型的Hello数据包中通告L2类型的优先级。与OSPF不同的是，在OSPF中如果路由器的优先级是0，那么它将没有资格成为一台指定路由器，而一台优先级为0的IS-IS路由器仅仅代表最低的优先级，但依然能够成为DR路由器。对于非广播型网络上的接口，不需要选举指定路由器，因此也将它们的优先级设置为0（注意示例10-1中显示的串行接口的优先级）。拥有最高优先级的路由器将成为指定路由器。如果路由器的优先级相同，那么拥有在数值上具有最高的MAC地址的接口的路由器将成为一台指定路由器。

与L1和L2类型的优先级相对应的是，需要在一个网络上为L1和L2分别选取单独的指定路由器。这种做法是必需的，因为在一个单一的局域网中存在着各自不同的L1和L2类型的邻接关系，如图10-7所示。由于一个接口对于每一层都具有单独不同的优先级，因此在同一个局域网上的L1类型的DR路由器和L2类型的DR路由器可能是同一台路由器，也可能不是。

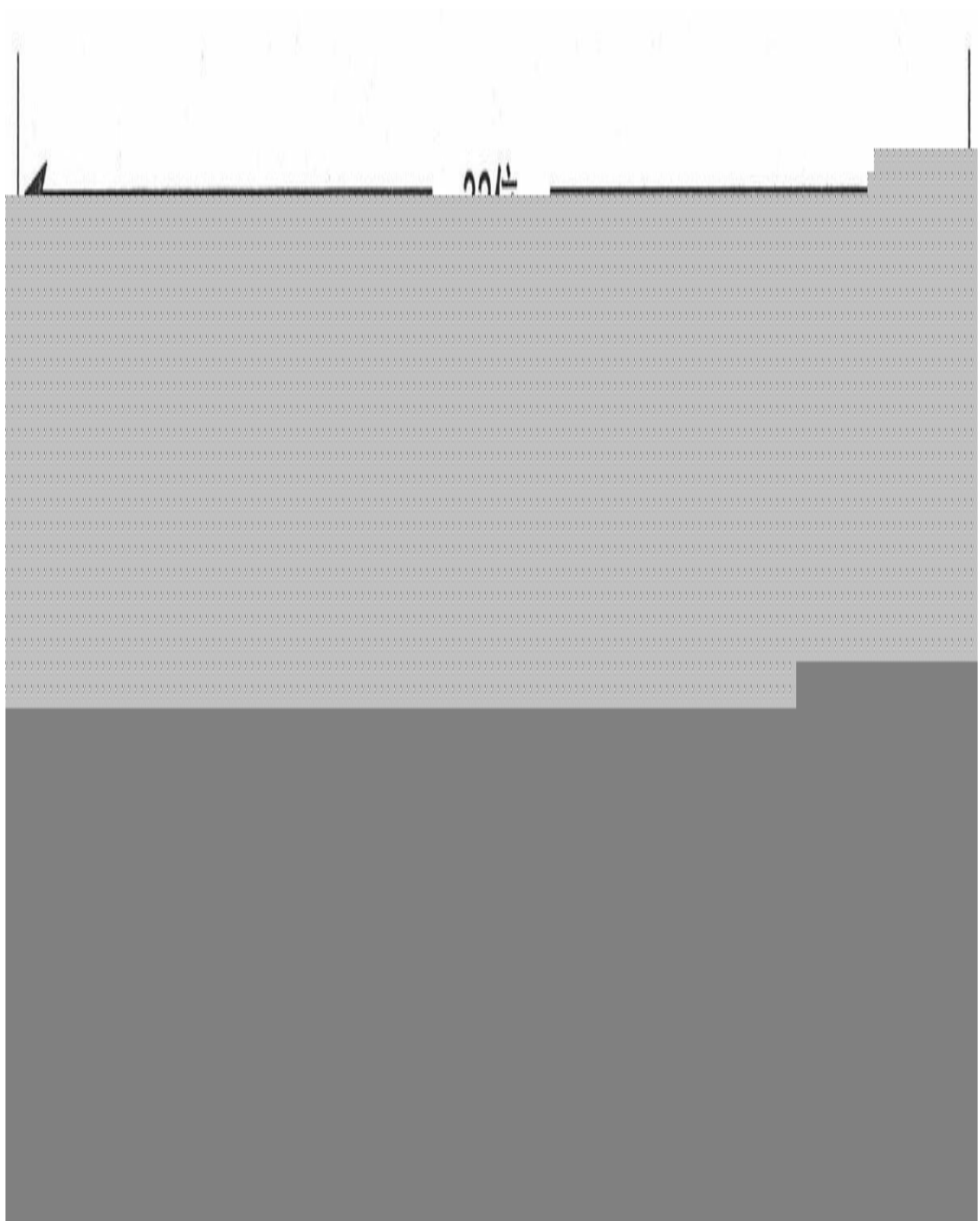
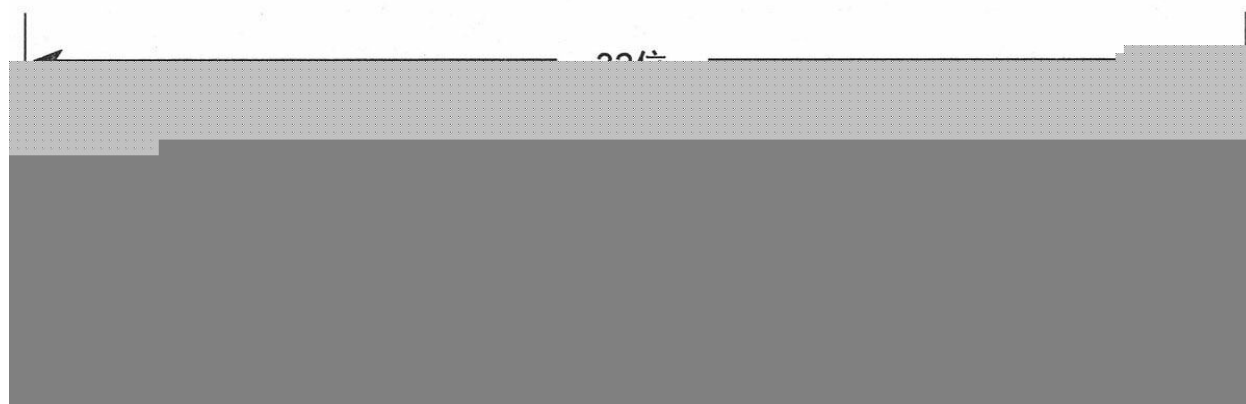


图10-7 对于第1层和第2层建立的邻接关系是不同的，因而也就必须为第1层和第2层选取各自的指定路由器

指定路由器分配了所在网络的LAN ID。正如前面章节所讨论的，LAN ID是由该网络上的指定路由器的系统ID和它的伪节点ID连在一起得到的。该网络上其他所有的路由器都将使用指定路由器分配的这个LAN ID。

如示例10-2所示，图中显示了一台路由器的E0接口的邻居表，这个E0接口是和示例10-1中显示的路由器的E0接口连接在同一个网络上的。通过比较这两个邻居表，可以发现总共有3台路由器连接在这个以太网上：0000.0c0a.2aa9、0000.0c0a.2c51和0000.0c76.5b7c。由于它们所有的优先级都是64，因此在数值上具有最高的MAC地址的路由器将成为指定路由器，也就是路由器0000.0c76.5b7c，并且它在这里利用设置为02的电路ID来标识这个网络。因此，示例10-1和示例10-2中显示的LAN ID都是0000.0c76.5b7c.02。

示例10-2 这台路由器的E0接口是和示例10-1中的路由器的E0接口连接在同一个网络上的



那个附加在系统ID后面的电路ID是必需的，因为同一台路由器可以是多个网络的指定路由器。注意在示例10-1中，与那台路由器的E0接口和E1接口相连的两个网络的指定路由器都是同一台路由器。与E0接口相连的网络的电路ID设置为02，而与E1接口相连的网络的电路ID设置为03，这样就可以使每一个网络的LAN ID保持惟一性。

IS-IS协议的指定路由器处理过程相比OSPF协议的指定路由器处理过程来说，有两个方面是十分粗糙的（或者说是不够复杂的，这个观点因人而异）。首先，IS-IS协议不选取备份指定路由器。如果IS-IS的指定路由器失效了，那么将选取一台新的指定路由器。其次，IS-IS的指定路由器相对OSPF的指定路由器来说是不稳定的。如果一台OSPF路由器在一

个已经存在指定路由器的网络上变成活动的了，即使它的优先级或路由器ID更高，新的路由器也不会成为一台指定路由器。结果是，OSPF的指定路由器通常是网络上处于活动状态很长的路由器。与OSPF的规则相比，如果一台新的IS-IS路由器具有比现有指定路由器更高的优先级，或者优先级相同但是具有更高的MAC地址，那么这台新的IS-IS路由器将成为新的指定路由器。这样，每次指定路由器的更改都必须有一组新的LSP数据包进行泛洪扩散。

但是，假设增加一台新的路由器到一条广播链路上对于一个正在运行的网络来说是相对偶然情况，并且DR的选举过程在现代路由器中处理的非常快（一般在微秒级），那么缺少BDR与固定选取DR的情况就可以不予考虑。

2. 独立于子网的功能

子网独立子层的功能定义了CLNS怎样分发数据包通过整个CLNP的网络，以及怎样把这些服务提供给传输层。路由选择功能本身又被分成4个过程：更新过程、决策过程、转发过程和接收过程。正如后面两个过程的名称所暗示的，转发过程（forwarding process）的职责是传送PDU数据包，而接收过程（receive process）的职责是接收PDU数据包。这两个过程是在ISO 10589中描述的，并且相比IP数据包来说，它们和CLNS NPDU数据包的关系更密切，因此不再做进一步的讲述。

（1）更新过程

更新过程（update Process）的职责是构建L1和L2的链路状态数据库。为了做到这一点，L1的LSP将在整个区域内进行泛洪，而L2的LSP将会在所有L2的邻接上进行泛洪。关于LSP数据包的具体字段的详细描述将在后面的10.1.4小节中讲述。

每一个LSP数据包都包含一个剩余生存时间、一个序列号和一个校验和。剩余生存时间（remaining lifetime）是一个老化时间或使用期限（age）。IS-IS LSP数据包的剩余生存时间和OSPF协议中LSA数据包的老化时间参数的一个不同是：LSA数据包的老化时间是从0到最大生存时间依次递增，而LSP数据包的剩余生存时间则是从最大生存时间开始，并递减到0。在这里，IS-IS的最大生存时间（MaxAge）是1200s（20min）。像OSPF协议一样，当LSP驻留在路由器的链路状态数

数据库中时，IS-IS会随着时间的推移老化每一个LSP，即递减它的剩余生存时间。并且，始发路由器必须周期性地刷新它的LSP以防止它的剩余生存时间减小到0。IS-IS的刷新时间间隔是15min减去一个最大不超过25%的随机抖动变量。如果剩余生存时间减小到0了，那么这个过期的LSP将还会在路由器的链路状态数据库中保留60s的时间，这个时间称为“零老化生存时间”（ZeroAgeLifetime）。

与OSPF LSA的刷新过程相比，一个有趣而重要的不同是，IS-IS的剩余生存时间由一个非0的数字开始，并递减到0，这里非0的数字缺省是1200s并能够更改。事实上，它可以增大到65535s——这大约是18.2小时。这在一个非常大型的区域中对IS-IS的扩展性是一个重要而有帮助的因素。如果区域的链路相当稳定，那么增大刷新时间和剩余生存时间可以显著地减少LSP刷新引起的泛洪扩散负载。在本地始发的LSP中设置的初始剩余生存时间（MaxAge）值可以通过命令max-lsp-lifetime更改，本地始发的LSP中刷新之间的周期可以通过命令lsp-refresh-interval设置。如果改变缺省值，读者应该把设置的刷新时间间隔至少比剩余生存时间少几百秒，以便在原来的LSP实例过期之前，使新的刷新LSP可以到达区域内的所有路由器。

在原来的IS-IS过程中，如果一台路由器收到了一个带有错误校验和的LSP，那么这台路由器可以通过将LSP的剩余生存时间设置为0并重新扩散出去来清除这条LSP。清除的行为将会引起始发这条LSP的路由器发送一个新的关于这条LSP的实例（instance）。但是，ISO 10589的第二版中将不再清除带有错误校验和的LSP，因为在一个可能出现错误的子网中，允许接收路由器启动清除LSP的功能会显著地增加LSP的流量。

在12.0版之前的IOS软件实现中也遵循这个旧的清除过程，但是可以通过命令ignore-lsp-errors来忽略这个行为。当一台启动了该选项的路由器收到一条被破坏的LSP时，它就会丢弃它而不是清除它。但是，这条被破坏的LSP的始发路由器仍然会利用SNP了解到这条LSP没有被收到。SNP将在本节后面的部分讲述。在IOS 12.0版本中实现了新的IS-IS过程，缺省情况下启动ignore-lsp-errors。

然而，有关LSP清除的一个重要特点，也是IS-IS区别于OSPF协议的另一方面，是在OSPF协议里只有始发路由器才能清除这个LSA。

序列号是一个32位的无符号线性数字。当一台路由器开始始发一条LSP时，它将使用设置为1的序列号，并且这条LSP的每一个后续实例的序

列号都会递增1。如果序列号递增达到了最大值（0xFFFFFFFF），那么这个IS-IS进程必须失效至少21min（最大生存时间+零老化生存时间），以便允许这条旧的LSP从所有的链路状态数据库中清除掉。

在一个点到点的子网上，路由器将直接发送L1和L2的LSP给它们的邻居路由器。在一个广播型的子网上，LSP将以组播的方式发送到它所有的邻居路由器。携带L1 LSP的帧会有一个0180.c200.0014的目的MAC标识，称为AllL1ISs。携带L2 LSP的帧会有一个0180.c200.0015的目的MAC标识，称为AllL2ISs。

IS-IS协议使用序列号数据包（SNP）来了解LSP的接收情况和维护链路状态数据库的同步情况。在这里有两种类型的SNP：部分序列号报文（PSNP）和完全序列号报文（CSNP）。在一个点到点的子网上，路由器使用PSNP来明确地确认每一个LSP数据包是否收到。[\[15\]](#) PSNP数据包是通过下面的信息来描述正在被确认的LSP：

- LSP ID;
- LSP的序列号;
- LSP的校验和;
- LSP的剩余生存时间。

当一台路由器在一个点到点的子网上发送一条LSP时，它会设置一个周期为minimumLSPTransmissionInterval的计时器。如果该计时器超时，路由器还没有收到一个关于确认收到这条LSP的PSNP数据包，那么将会发送一个新的LSP数据包。在Cisco路由器上，minimumLSPTransmissionInterval的缺省值是5s，并且可以基于每接口使用命令**isis retransmit-interval**来更改。缺省值通常几乎都是合适的值，偶尔的例外情况是某个邻居可能超负荷地处理大量的LSP数据包。这种情况下，路由器可能无法足够快地确认LSP，从而触发一个重传数据包，并增加了它的故障。在这种情况下，增大minimumLSPTransmissionInterval是有帮助的；但最终真正解决这种问题的办法是升级低性能的邻居路由器。

在一个广播型子网上，LSP不需要每一台接收它的路由器确认。作为替代，指定路由器将会周期性地以组播方式发送CSNP，用来描述链路状

态数据库中的每一个LSP。发送CSNP的周期缺省的是10s，并且可以通过命令**isis csnp-interval** 来更改。L1 CSNP以组播方式发送到AllL1ISs（0180.c200.0014），而L2 CSNP以组播方式发送到AllL2ISs（0180.c200.0015）。

当一台路由器收到一个CSNP时，它会把PDU中的LSP摘要与自己数据库中的LSP进行比较。如果发现在该路由器的数据库中存在CSNP中没有的LSP，或者存在比CSNP中更新的LSP实例，那么该路由器将以组播方式在网络上发送这条LSP。但是，如果其他的路由器开始发送更新的LSP，那么该路由器将不会发送相同LSP的另一个拷贝。如果路由器的数据库中没有包含CSNP中列出的所有LSP，或者数据库中包含的是某条LSP的旧实例，那么这台路由器将会以组播方式发送一个PSNP数据包，这个数据包中列出了该路由器所需要的所有LSP。虽然PSNP数据包是以组播方式发送的，但是只有指定路由器才会使用包含相应LSP的数据包来响应。

IS-IS具有一个有趣的特性，如果运行它的设备因内存不足而不能记录完整的链路状态数据库时，它具有通知其他路由器的能力。导致内存溢出或过载的原因可能是因为该路由器所在的区域变得过于庞大，路由器的内存不足，或者一些瞬间情况——像指定路由器失效等。在上面这些情况下，路由器如果不能完整地存储链路状态数据库，那么它将会在所发送的LSP数据包中设置一个称为过载（Overload, OL）的位。

OL位用来指示路由器可能不能再进行正确的路由选择决策了，因为它的链路状态数据库已经不再完整。其他的路由器将仍然会转发数据包到这台过载路由器的直连网络上，但是，在这台过载的路由器发送一个清除OL位的LSP数据包之前，其他路由器不会再利用这台路由器转发经过它传送的数据流了。因为设置OL位可以避免过载的路由器被用作一条路由的下一跳，因此这个位又经常被称做hippity位（或称为hippity-hop，随个人习惯叫法）。一般来说，路由器的内存应该平等地分配给L1和L2的数据库，但是，路由器能够在其中一个层的内存过载时，而保持其他层的内存处于正常状态。

过载特性是在路由器没有那么多内存时的产物。现代的路由器一般不太可能出现过载的情况。但是，OL位在现代的网络中，特别是在BGP网络中还有另外一个非常有用的用途。为了了解这个问题，有必要先了解一些BGP协议的基本概念。虽然像IS-IS这样的链路状态协议收敛速度非常快，但BGP网络的收敛却比较慢——有时需要花上几分钟的时间。在

一个BGP网络的内部，路由可能具有距离好几跳路由器的下一跳地址。当一个数据包到达时，将在BGP路由中查找它的目的地址。转发这个数据包之前，在IGP的路由表中必须能够查找到该BGP路由的下一跳。

现在假定有一台新路由器增加到某个网络上，并且IGP已经被收敛但BGP还没有完成收敛。如果另一台路由器根据收敛的IGP路由确认这台新增加的路由器是到达BGP下一跳的最优路径，那么它将把向下一跳地址转发的数据包转发到这台路由器上。但是，如果BGP还没有在新的路由器中完成收敛，该路由器就可能没有关于这个数据包的BGP路由，从而丢弃该数据包，结果变成路由业务黑洞。

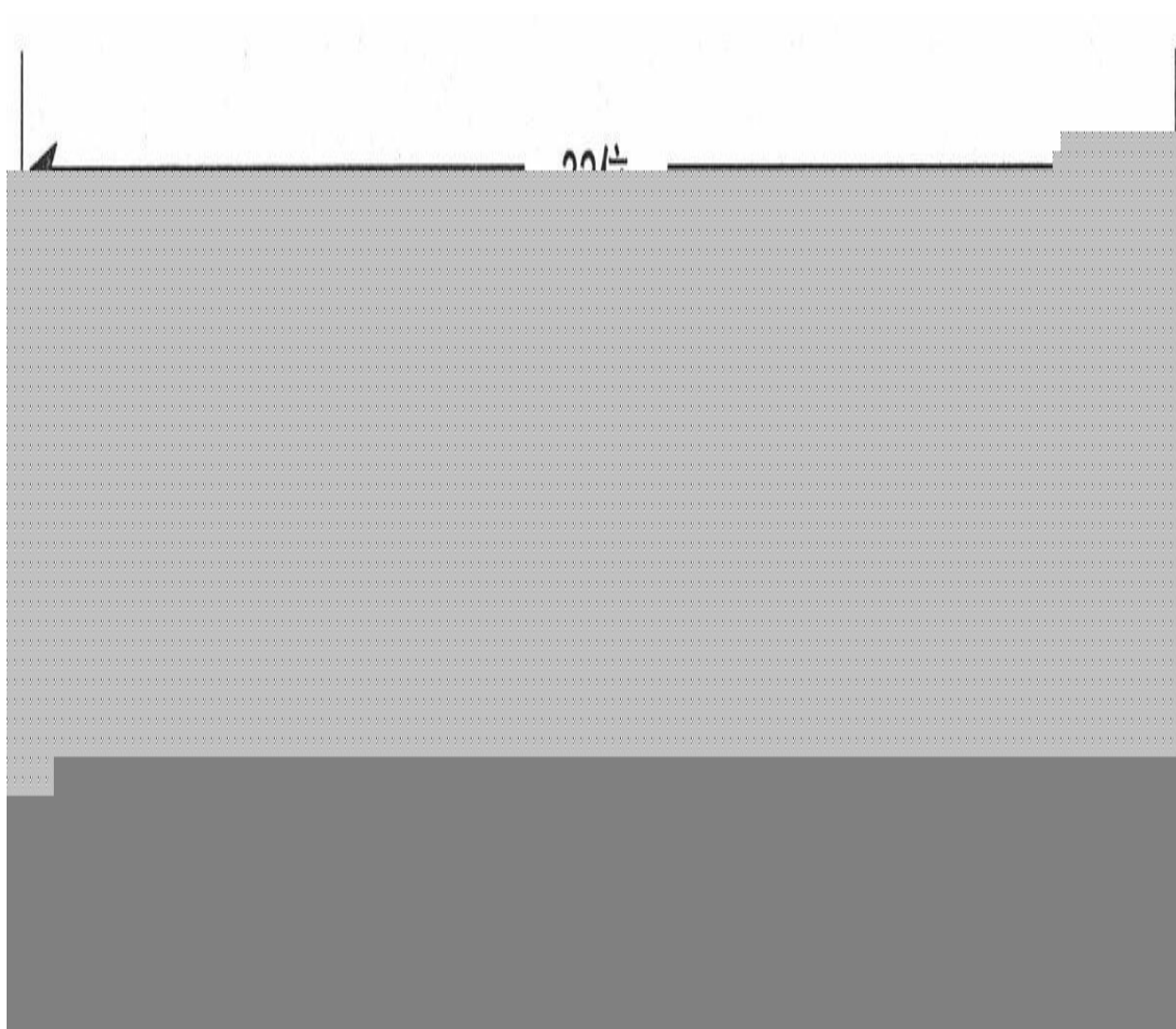
在BGP完成收敛之前通过设置IS-IS的OL位，我们就可以避免这个问题。如果设置了OL位，那么其他路由器将会绕过这台新的路由器进行路由选择。一旦BGP完成收敛，OL位将被清除，数据包将可以通过这台新路由器进行转发。

使用命令**set-overload-bit on-startup** 可以指定一个秒数，用来说明在IS-IS启动后需要设置OL位的时间。当超过指定的秒数时，OL位会被自动地清除。将OL位设置为大约300~500s，可以确保BGP在OL位被清除前完成BGP的收敛。这里有另一个可选的命令**set-overload-bit on-startup wait-for-bgp**，这样一旦BGP完成收敛就会清除OL位。虽然这比使用一个静态的秒数更可以动态地调整，但这种做法也有一个致命的缺点：如果某个BGP会话因为某些原因无法形成，那么OL位将永远都不会被清除。因此，通常的做法最好还是指定一个秒数代替**wait-for-bgp** 选项。

我们也可以通过使用不指定其他选项的**set-overload-bit** 手工地模糊设置OL位。这在我们希望把一台路由器连接到IS-IS网络上并接收全部的数据库，但又不希望这台路由器用来作为转发路径时比较有用。这种应用的一个例子就是，把一台试验的路由器连接到一个商用网络的情形。

使用命令**show isis database** 可以显示一个IS-IS链路状态数据库的摘要，如示例10-3所示。其中显示了路由器Brussels是一台L1/L2路由器，因此它包含了L1和L2的数据库。在第一列显示的LSP ID是由始发路由器的系统ID连接了2个八位组字节构成的。在这里，跟在系统ID后的第一个八位组字节是伪节点ID。如果这个八位组字节是非零的，LSP则是由一台DR路由器始发的。这时，系统ID和非零的伪节点ID一起构成了一个广播型子网的LAN ID。

示例10-3 IS-IS的数据库可以通过命令**show isis database**来查看



LSP ID的最后一个八位组字节是LSP的编号。有时候一个LSP可能会很大，以至于超出了路由器缓冲区或数据链路所支持的MTU的大小。在这种情况下，LSP将被分段传送——也就是说，LSP的信息可以在多个LSP数据包中传送。这些LSP数据包的LSP ID由3部分组成：相同的系统ID和伪节点ID，还有不同的LSP编号。

LSP ID后面紧跟的星号，表示这条LSP数据包是始发于正在查看的数据库所在的路由器。例如，在示例10-3中显示的数据库是来自路由器Brussels的。因此，LSP ID为0000.0c76.5b7c.00-00的L1 LSP是始发于路由器Brussels的。

数据库的第2列和第3列显示了每一个LSP的序列号和校验和。第4列显示的是LSP抑制时间，也就是LSP的剩余生存时间，以秒数计。如果连续不断地重复输入**show isis database**命令，就会发现这个数字在不断减小。当重新刷新一条LSP时，LSP的剩余生存时间将被重新设置为1200s，并将序列号递加1。

最后一列指明了每一个LSP的区域关联位（ATT位）、区域分段位（Partition, P位）和过载位（OL位）。L2和L1/L2路由器设置ATT位为1来指明它们含有到达其他区域的路由。P位指出始发路由器具有支持区域分段修复的能力。Cisco公司（和大多数其他厂商）并不支持这个功能，因此该位总是设置为0。OL位设置为1，可能是始发路由器正处于内存过载状态，而这时链路状态数据库是不完整的，或者是被手工设置的。

参见示例10-4所示，使用带level和LSP ID的**show isis database detail**命令可以查看一个LSP的完整信息。LSP中每一个单独字段的具体含义参见10.1.4小节。

示例10-4 使用命令**show isis database detail**可以查看数据库中LSP的完整信息

（2）决策过程

一旦更新过程建立了链路状态数据库，决策过程就将使用数据库中的信息去计算一个最短路径树。接着，该过程使用生成的最短路径树去构建一个转发数据库（路由表）。对于L1数据库和L2数据库，路由器将会执行不同的SPF计算。

ISO 10589规定IS-IS协议使用下面的度量（一项是必须的，三项是可选的）来计算最短路径：

- 缺省度量（**Default**）——这是每一台IS-IS路由器都必须支持和理解的度量。
- 时延度量（**Delay**）——这是一个可选项，反映一个子网的传输时延。

- 代价度量（**Expense**）——这是一个可选项，反映一个子网的成本代价。
- 差错度量（**Error**）——这是一个可选项，反映一个子网的出错概率，类似于IGRP/EIGRP协议中的可靠度量。

每一种度量都使用一个范围在0~63之间的整数表示，并且每个路由都要为每种度量进行单独地计算。因此，如果一个系统同时支持这4种度量类型，那么路由器必须为L1数据库和L2数据库各运行4次SPF计算。由于对于每一个目的的路由都可能需要进行多次反复地计算，结果会产生多个不同的路由表；因为可选的度量是用来支持根本没有发展起来的服务类型（ToS）的路由选择的，因而Cisco公司只支持缺省度量。

在Cisco的路由器上，不论接口的类型如何，都会指定每一个接口的缺省度量为10。使用命令**isis metric**可以修改这个缺省度量的值，而且可以分别为第1层和第2层的接口修改它们的缺省值。如果对于每一个接口都保留使用它的缺省度量10，那么每个子网的度量都可以被认为是等价的，并且每个子网的IS-IS度量可以看作是一个简单的跳数，其中每一跳的代价为10。

这种情况下，一条路由的总代价就可以看作是沿此路由路径方向的每一个出站接口的单独度量简单相加。对于任何一条路由，IS-IS最大的度量值是1023。这个比较小的最大度量值经常被认为是IS-IS协议的一个限制，因为在一个大型的网络上它的度量尺度显得有点小了。

但是，IS-IS中新的扩展允许非常大的度量空间，具有32位度量，称为扩展度量（**wide metrics**）。使用命令**metric-style wide**可以设置扩展度量。

IS-IS协议的路由不仅分L1路由和L2路由，而且分内部路由和外部路由。内部路由是指到达IS-IS路由选择域内的目的地的路径，而外部路由是指到达IS-IS路由选择域外的目的地的路径。虽然L2路由可能是内部路由，也可能是外部路由，但L1路由却总是内部路由。使用路由选择策略我们能够把一条L1路由变成外部路由，但是只有在比较合理和比较小心的情況下才应该这样做。

如果到达某个具体的目的地存在多条可能的路由，那么L1的路由将优先于L2的路由。在同一层的多条路由中，支持可选度量的路径要优先于只

支持缺省度量的路径（再次提示，Cisco的路由器仅仅支持缺省度量，因此第二个优先顺序的排序和Cisco的路由器不相关）。在每一层所支持的度量中，具有最低度量的路由优先。如果经过这个决策处理后发现多条路径在同一层是等价的，那么它们都会被放入路由表中。在Cisco公司的IS-IS协议的实现中将执行等代价的负载均衡，并且最大支持6条等价负载均衡的路径。

在前面“更新过程”中谈到LSP ID的最后一个八位组字节是LSP编号（LSP Number），并用来跟踪分段的LSP。决策过程关注这个LSP编号有几种原因。首先，如果一个LSP编号为0并且剩余生存时间不为0的LSP不在路由器的数据库中，那么决策过程将不会处理任何具有同样的系统ID但LSP编号不为0的LSP。例如，假设在数据库中存在LSP ID为0000.0c76.5b7c.00-01和0000.0c76.5b7c.00-02的两条LSP，但是在该数据库中没有包含LSP ID为0000.0c76.5b7c.00-00的LSP，那么路由器将不会处理前面的两条LSP。这种方法可以确保不会因不完整的LSP而导致不精确的路由选择决策。

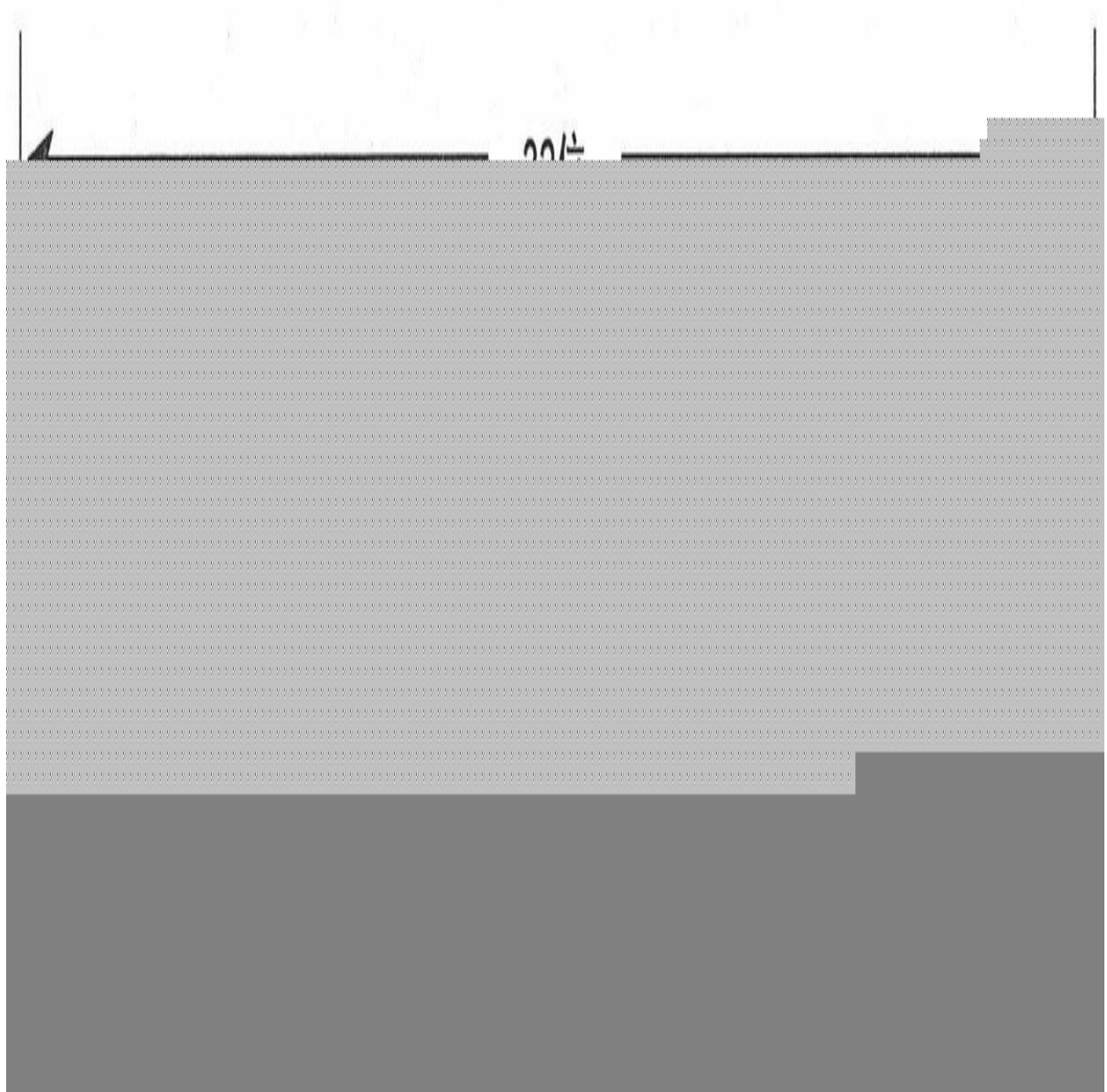
另外，决策过程将仅从LSP编号为0的LSP中接受下面的信息：

- 数据库中过载位的设置信息；
- IS类型字段的设置信息；
- 区域地址可选字段的设置信息。

但是在LSP编号不为0的LSP中，决策过程将忽略这些设置信息。换句话说，在一系列被分段的LSP中，将由第一个LSP来宣告所有分段LSP的这3个设置信息。

参见示例10-5，图中显示了一台Cisco的IS-IS路由器的路由表。在这里我们注意到存在L1和L2的路由，并且有3个到达目的地的路由具有多条路径。每一条路由都带有一个相应的掩码，这表明它们是支持VLSM技术的。最后，在路由表中也指出了IS-IS路由的管理距离是115。

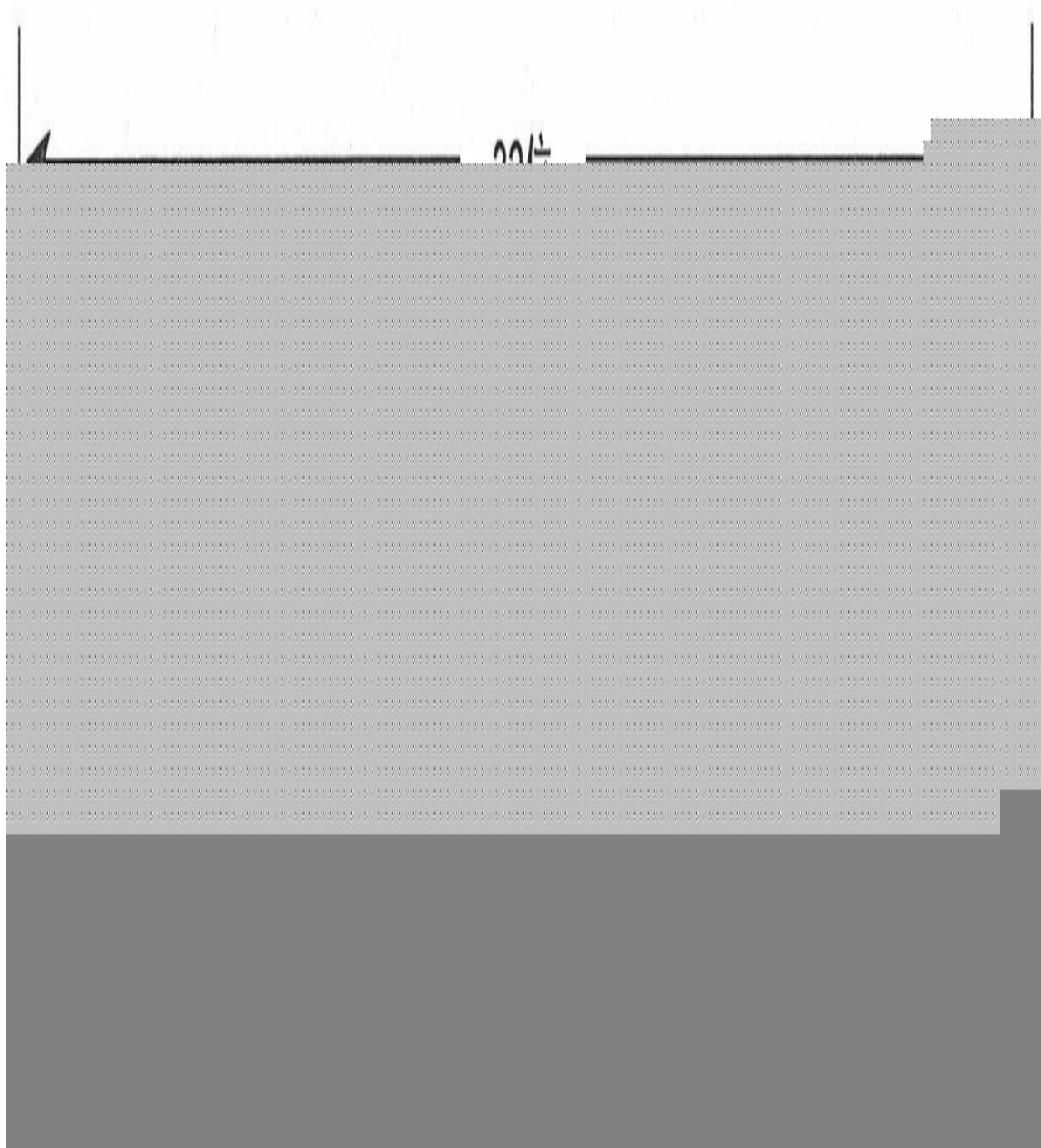
示例10-5 这个路由表显示了第1层和第2层IS-IS路由



对于L1的路由器来说，决策过程还有另外一项功能，就是为区域间路由选择计算到达最近的L2路由器的路径。正如前面所提到的，当一台L2或L1/L2路由器与其他区域相连时，路由器将通过在它的LSP中设置ATT位为1来通告这种情况。对于L1路由器，决策过程将选择度量最近的L1/L2路由器作为它缺省的区域间路由器。当使用IS-IS协议进行IP路由选择时，在路由器中会记录一条到达L1/L2路由器的IP缺省路由。例如，在示例10-6中显示了一台L1路由器的链路状态数据库和相应的路由表。LSP0000.0c0a.2c51.00-00的ATT位设置为1。基于这个信息，决策过程将选择系统ID为0000.0c0a.2c51的路由器作为缺省的区域间路由器。

在路由表中还显示了一条经过10.1.255.6可达的缺省路由（0.0.0.0），它的度量是10。虽然在示例10-6的两个表中显示的信息不太容易对照，但实际上地址为10.1.255.6和系统ID为0000.0c0a.2c51.00-00的路由器指的是同一台路由器。

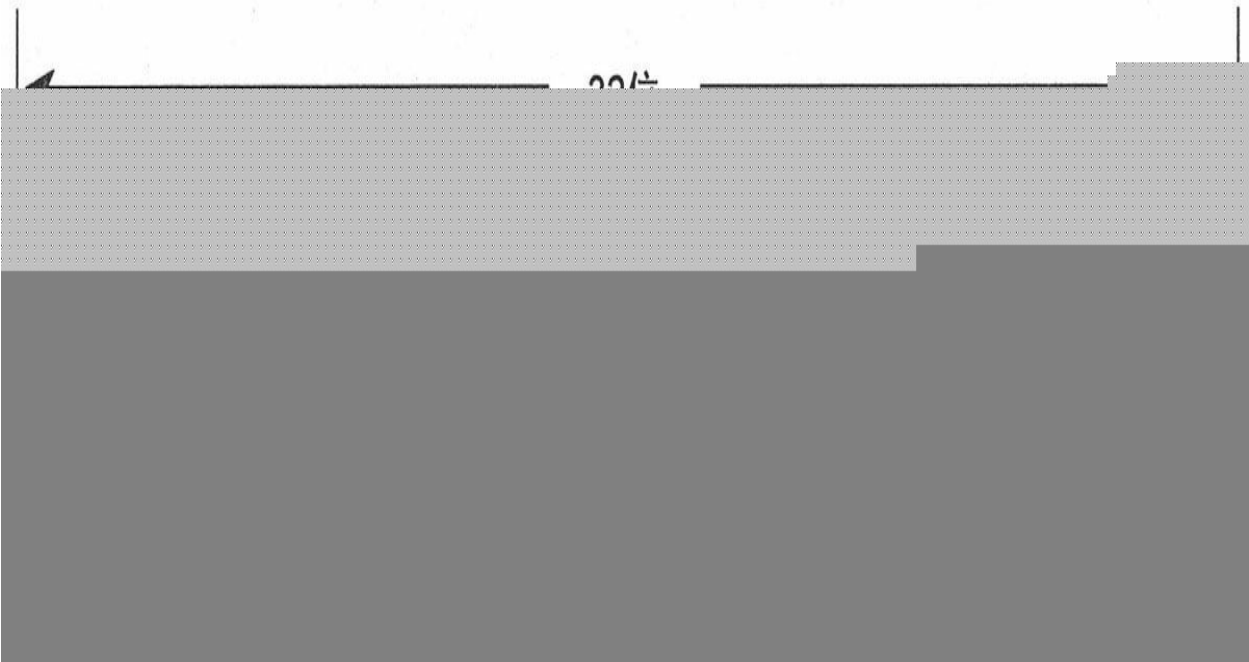
示例10-6 如果**ATT**位设置为**1**，集成**IS-IS**就会增加一条到达最近的**L1/L2**路由器的**IP**缺省路由



示例10-6中显示的信息突出了采用IS-IS协议的一个有趣特性，特别是在进行故障排除时就更加突出。虽然TCP/IP协议是被路由转发的协议，但是决定路由选择策略的协议，包括所有路由的控制数据包和地址却都是CLNS协议。有时要把CLNS协议的信息和IP协议相关的信息关联起来是比较困难的。解决该问题的，有一条很有用的命令是**which-route**。

这条命令主要用来确定某个具体的CLNS目的地址在路由表的定位。但是，利用**which-route** 命令也可以了解到一些关于某个具体CLNS地址和相关IP地址的有用信息。如示例10-7所示，显示了使用系统ID / 电路标识0000.0c0a.2c51.00作为参数的**which-route** 命令的输出信息，这里的系统ID0000.0c0a.2c51.00指的就是在前面示例10-6显示的数据库中ATT=1的那个LSP。从输出结果可以看出，要查询的系统ID的下一跳IP地址是10.1.255.6。

示例**10-7** 使用**which-route**命令可以发现一些CLNS地址和IP地址的关联信息



10.1.4 IS-IS的PDU格式

IS-IS协议使用9种PDU类型来进行它的控制信息处理，并使用一个5位的类型号来标识每一个PDU数据包。所有的PDU数据包可以归纳为3类，如表10-1所示。

在所有IS-IS PDU数据包起始的8个八位组字节都是该数据包的头部字段，并且对于所有的PDU数据包类型来说都是公用的，如图10-8所示。这里将先讲述这些起始字段，特有的PDU字段将在后续的章节讲述。

表10-1 S-IS协议的PDU数据包类型

IS-IS PDU	类型号
Hello PDU	
第 1 层 LAN 的 IS-IS Hello PDU	15
第 2 层 LAN 的 IS-IS Hello PDU	16
点到点的 IS-IS Hello PDU	17
链路状态PDU	
第1层LSP	18
第2层LSP	20
序列号PDU	
第1层完全序列号	24
第2层完全序列号	25
第1层部分序列号	26
第2层部分序列号	27

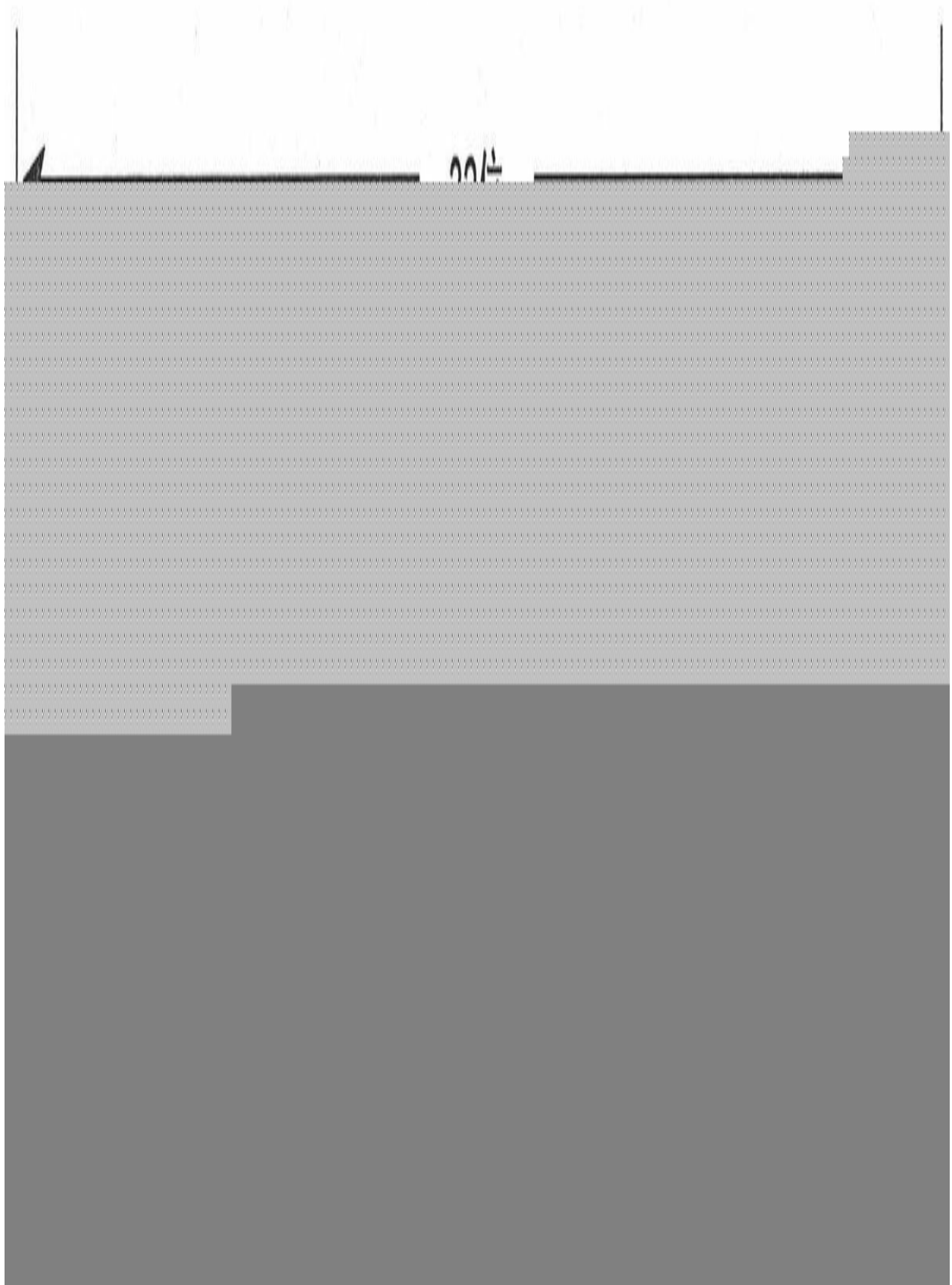


图10-8 IS-IS PDU数据包起始的8个八位组字节

- 域内路由选择协议鉴别符（**Intradomain Routeing Protocol Discriminator**）——这是由ISO 9577 [\[16\]](#)分配的一个恒定不变的数值，用来标识网络层协议数据单元（NPDU）。在所有的IS-IS PDU中，该字段的值都是0x83。
- 长度标识符（**Length Indicator**）——标识该固定头部字段的长度，以八位组字节数表示。
- 版本 / 协议ID扩展（**Version/Protocol ID Extension**）——当前始终设置为1。
- ID长度（**ID Length**）——用来标识该路由选择域内使用的NSAP地址和NET的系统ID（System ID）的长度。该字段的取值可以是以下几个数值之一：
 - ☐ 1~8的整数，表示系统ID字段具有相同长度的八位组字节数；
 - ☐ 0，表示系统ID字段的长度为6个八位组字节；
 - ☐ 255，表示系统ID字段为空（0个八位组字节）。

在Cisco路由器中系统ID字段的长度固定为6个八位组字节，因此，在由Cisco路由器始发的PDU数据包中，这个ID长度字段的值将始终是0。

- **PDU类型** ——是一个5位的字段，取值范围可以是表10-1中显示的PDU类型号中的任何一个。该字段的前3位（R）作为保留位，始终为0。
- 版本号（**Version**）——当前始终设置为1，这和第3个八位组字节中的版本 / 协议ID扩展字段是一致的。
- 保留位 ——当前设置为全0。
- 最大区域地址数（**Maximum Area Addresses**）——表示该IS区域所允许的最大区域地址数量。这个字段的值可以是下面数值之

一：

- ☐ 1~254的整数，表示该区域实际所允许的最大区域地址数；
- ☐ 0，表示该IS区域最多只支持3个区域地址数。

Cisco IOS软件缺省情况下最多支持3个区域。因此，在由Cisco路由器始发的IS-IS PDU 中，该最大区域地址数字段的值始终是0，除非通过命令 **max-area-addresses** 改变缺省配置。

在图10-9中，显示了使用协议分析仪捕获到的某个IS-IS PDU起始的8个八位组字节。紧跟在公共头部字段之后的专有PDU字段也是头部的一部分。根据PDU类型的不同它们会有所变化，这将在讲述具体PDU类型的章节中介绍。

img483_1

图10-9 使用协议分析仪捕获到的某个IS-IS PDU起始的8个八位组字节

1. TLV字段

紧跟在专有PDU字段之后的可变长度字段是类型 / 长度 / 值 (Type/Length/Value, TLV) [17] 这3个参数的不同组合，如图10-10所示。类型（或代码） [18] 是一个指定值字段信息类型的数字，长度用来指定值字段的长度，而值字段就是它本身的信息内容。注意，长度字段只用一个八位组字节来表示，这意味着值字段的最大值为255个八位组字节。



图10-10 IS-IS的TLV参数的组合在IS-IS协议中的功能和TLV的组合在EIGRP协议中的功能相同

表10-2中列出了所有IS-IS协议里TLV的代码。在这个表中也指出了哪些TLV是在ISO 10589中指定的，哪些是在RFC 1195中指定的。ISO指定的TLV是设计用于CLNP协议的，但它们中的大多数也用于IP协议。RFC指定的TLV只设计用于IP协议。如果一台路由器不能识别一个特有的TLV代码，那么它将忽略这个TLV。这种设计方法可以允许在同一个PDU中携带支持CLNP协议的TLV、支持IP协议的TLV，或者同时携带这两种TLV。

表10-2

IS-IS协议中使用的TLV代码

img484_1

虽然大多数的TLV都在不止一种IS-IS PDU类型中使用，但是只有一个TLV（认证信息TLV）是在所有的PDU中都使用的。在下面讲述IS-IS PDU格式部分，将会列出每一种PDU用到的TLV。每一种TLV的格式将只在它第一次出现时讲述一次。表10-3中总结了每种PDU所用到的TLV。

表10-3

每一种IS-IS PDU所用到的TLV

img484_2

使用TLV的最大好处是我们只需要增加新的TLV类型，就可以为IS-IS添加新特性。自从开始使用TLV以来，IS-IS协议中已经增加了很多增强型特性。表10-4中列出了自从ISO 10589和RFC 1195规定IS-IS以来已经增加的许多TLV类型。表中同时也列出了RFC规定的新TLV和它们的用途。在一些实例中，扩展特性还非常新，在编写本书时它们还是Internet草案形式。但在读者阅读本章的时候，它们也许已经成为RFC了。这可以通过IETF的Web网站（www.ietf.org）查找。表10-4中列出的部分扩展特性在本章随后的一些章节会有更详细地描述。

img485_1

表10-4

TLV用于扩展IS-IS特性

2. IS-IS Hello PDU格式

IS-IS Hello PDU是同一条链路上的IS-IS路由器用来发现它们的IS-IS邻居路由器的。一旦路由器发现了它的邻居路由器并且成功建立邻接关系，Hello PDU的工作就只担当keepalive的功能，从而维护已有的邻接关系并将邻接关系中任何参数的变化通知邻居。

一个IS-IS PDU大小的上限可以由始发路由器的缓冲区大小，或者传输该PDU的数据链路的MTU值来确定。ISO 10589规定IS-IS Hello数据包必须使用类型8的填充TLV，来填充一个小于该最大值的八位组字节，以便使路由器可以和它的邻居路由器以它的MTU进行通信。更为重要的是，发送达到或接近链路MTU大小的Hello数据包可以帮助检测这样一种链路的失效情形：较小的PDU可以通过，但是较大的PDU数据包会被丢弃。如果一个邻居的接口不支持所要求的最小MTU，那么被填充的Hello将被丢弃，并无法建立邻接关系。

有两种类型的IS-IS Hello数据包：LAN Hello数据包和点到点Hello数据包。LAN Hello数据包可以进一步分为L1和L2的LAN Hello数据包。但是，这两种类型的LAN Hello数据包的格式是相同的，如图10-11所示。图10-12显示了协议分析仪捕获到的一个L2的LAN Hello数据包。

- **电路类型（Circuit Type）** ——是一个2位的字段（前面6位是保留位，始终设置为0），用来指定该路由器是L1路由器（01）、L2路由器（02），还是L1/L2路由器（11）。如果这两位都为0（00），那么这个PDU数据包整个都会被忽略。
- **源ID（Source ID）** ——是指始发该Hello数据包的路由器的系统ID。
- **抑制时间（Holding Time）** ——是指邻居路由器在宣告始发路由器失效之前，它所等待接收下一个Hello数据包的时间间隔。
- **PDU长度** ——是指整个PDU数据包长度的八位组字节数。
- **优先级** ——是一个用来选取DR的7位字段。这个字段可以设置成0~127之间的数值，而且数值越大就表示优先级越高。L1的指定

路由器是通过L1 LAN Hello数据包中的优先级特性选取出的，而L2的指定路由器是通过L2 LAN Hello数据包中的优先级特性选取出的。

- **LAN ID** ——就是指定路由器（DR）的系统ID再加上一个八位组字节（伪节点ID），用来区分这个LAN ID和同一台指定路由器上的其他LAN ID。

img486_1


The diagram area is mostly blank, suggesting the content of the diagram is not visible or is a placeholder.

图10-11 IS-IS LAN Hello PDU数据包的格式

在IS-IS LAN Hello数据包中可以使用下面的多种TLV： [\[19\]](#)

- 区域地址（类型1）；
- 中间系统邻居（类型6）；
- 填充（类型8）；
- 认证信息（类型10）；
- 可选的校验和（类型12）；
- 支持的协议（类型129）；
- IP接口地址（类型132）；
- 重启（类型211）；
- 多拓扑（类型229）；
- IPv6接口地址（类型232）；
- 实验用（类型250）。

img487_1

图10-12 通过协议分析仪捕获到的LAN Hello数据包显示了只有Hello PDU数据包才有的字段

如图10-13所示，显示了IS-IS点到点Hello PDU数据包格式。点到点Hello数据包的格式和LAN Hello数据包相比，除了没有优先级字段外基本上是一样的，而且它用本地电路ID字段替代了LAN ID字段。与LAN Hello数据包不同，L1和L2的信息是在同一个点到点Hello PDU数据包中传送的。除了中间系统邻居TLV，点到点Hello数据包可以携带所有相同的TLV。作为替代，如果IS-IS实现支持相关的扩展，点到点Hello数据包携带一个P2P三方邻接（类型240）的TLV。这个TLV的用途将在10.1.5小节有关“三方握手”部分讲述。

img487_2

图10-13 IS-IS点到点Hello PDU数据包的格式

- **本地电路ID（Local Circuit ID）** ——是一个1个八位组字节的标识字段，由始发路由器发送Hello数据包时分配给这条电路，并且在路由器的接口上是惟一的。在点到点链路的另一端，Hello数据包中的本地电路ID可能包含，也可能不包含同样的值。

（1）区域地址TLV

如图10-14所示，区域地址TLV是在始发路由器上配置的，并用来通告该区域的地址。正如多个地址长度 / 区域地址字段所表示的，一台路由器可以配置多个区域地址。

在图10-15中显示了一个IS-IS Hello PDU数据包的部分信息。“Variable Length Field #3”部分显示的就是一个区域地址TLV的信息，它总共有6个八位组字节长，包括两个区域地址：47.0002（3个八位组字节）和0（1个八位组字节）。

img488_1

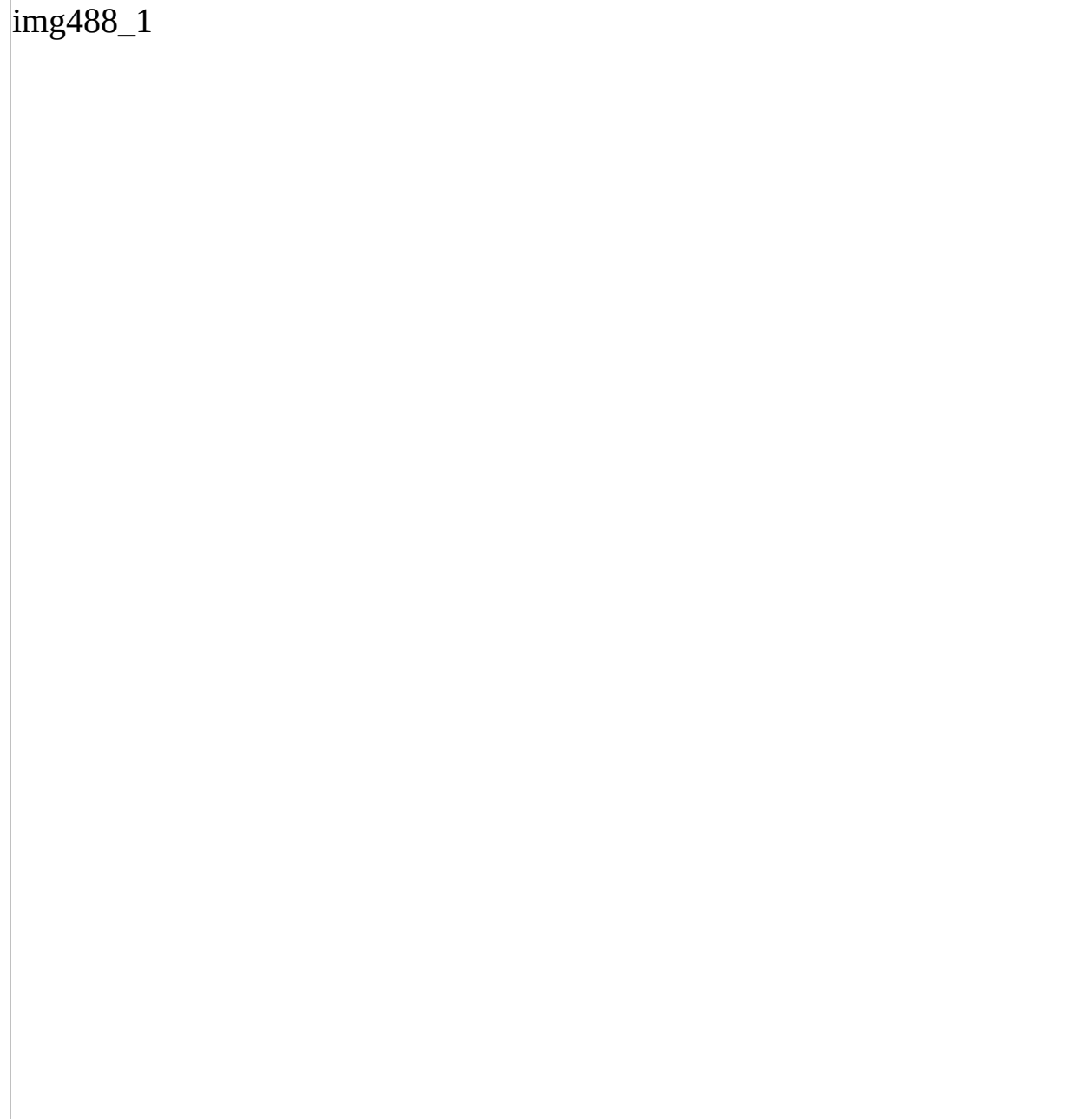


图10-14 区域地址TLV的格式

img488_2

图10-15 “Variable Length Field #3”部分显示的就是一个区域地址TLV的信息。为了便于观察，分析仪也在TLV中指出了所列出的所有地址数，但是这个信息并不是TLV字段的一部分

（2）中间系统邻居TLV（Hello）

如图10-16所示，中间系统邻居TLV列出了本地路由器所有邻居的系统ID，但是这些邻居路由器必须满足在最新的一个抑制时间间隔内，能够被本地路由器接收到它们发出的Hello数据包。这里可以注意到，这个TLV字段对于IS-IS LAN Hello数据包所起到的功能，其实在OSPF协议中也有相类似的功能：为了验证双向通信，本地路由器列出了最近发出的所有Hello数据包以及被其收到的邻居路由器。

这个TLV只能用在LAN Hello数据包中。由于点到点Hello数据包中没有指定路由器的选取过程，因而在点到点Hello数据包中没有这个TLV。同时，这里的TLV也和LSP数据包中使用中间系统邻居TLV有所不同，这可以通过它们的类型号码区分开来。L1 LAN Hello 数据包只列出了L1

的邻居，而L2 LAN Hello数据包也只列出了L2的邻居。在这个TLV中列出的邻居是由该LAN上它们接口的MAC地址标识的。图10-17中显示的“Variable Length Field #5”部分表示了一个中间系统邻居TLV，它只列出了一个邻居——0000.0c0a.2aa9。

img489_1

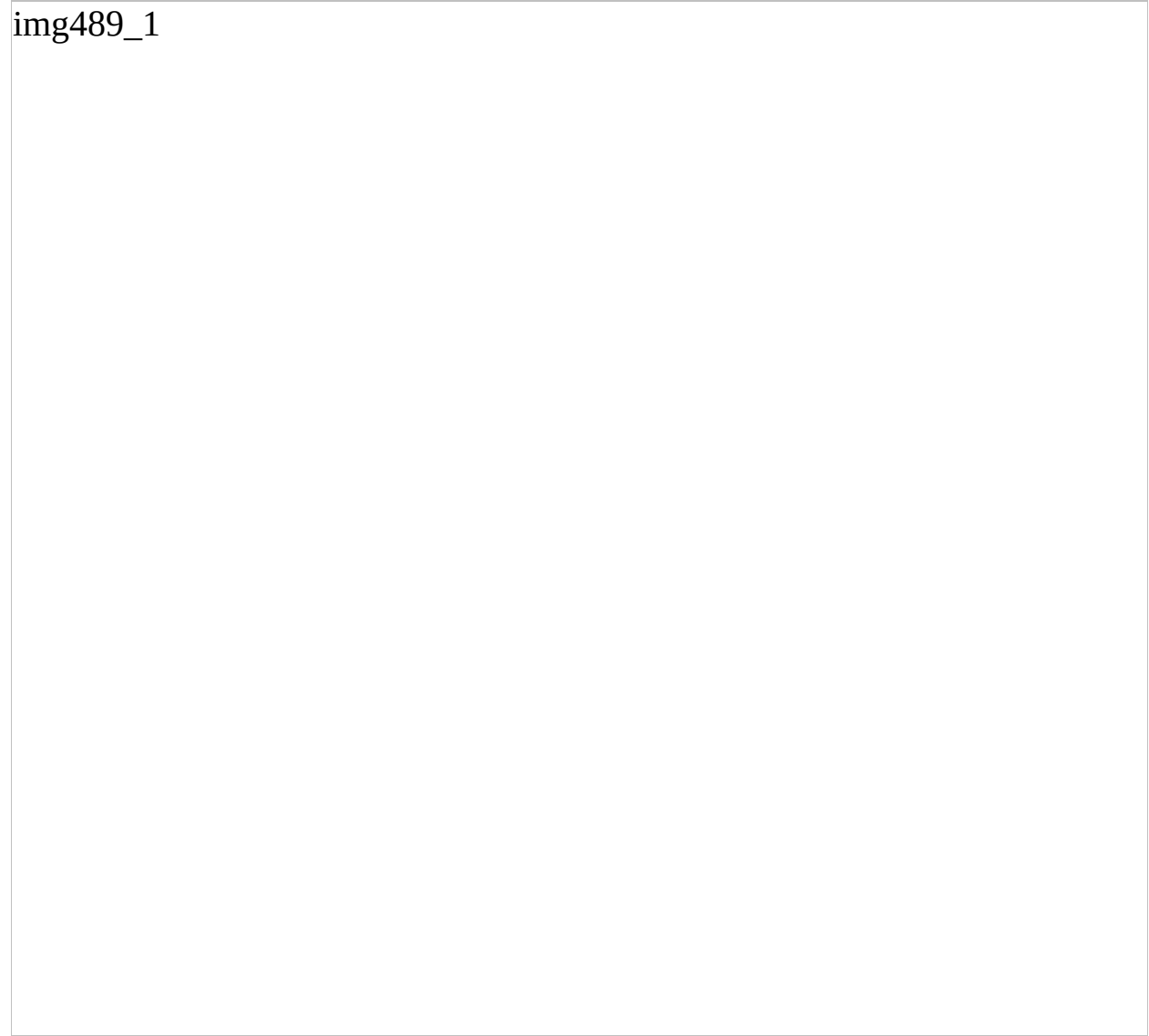


图10-16 Hello PDU数据包中的中间系统邻居TLV的格式

img489_2




图10-17 “Variable Length Field #5”部分显示了一个中间系统邻居TLV

（3）填充TLV

填充TLV是用来填充一个Hello PDU数据包，以使它达到允许的IS-IS大小的最小值（1492字节），或者该链路的MTU大小。由于一个值字段的最大长度为255字节，因此经常会使用多个填充TLV字段。由于填充TLV的内容被路由器忽略，因此在它的值字段内可以设置任意的数值。在Cisco路由器中这些位的值都被设置为0，如图10-18所示。

（4）认证信息TLV

如图10-19所示，认证信息TLV只有在配置认证时才会使用。认证类型字段包含了一个0~255之间的数字，用来指定认证使用的类型，并且也指定认证值字段所包含的认证信息的类型。IOS软件支持明文口令认证，或HMAC-MD5加密散列认证。明文口令认证是认证类型1，而HMAC-MD5是认证类型54。

如图10-15所示，“Variable Length Field #1”部分显示的就是一个认证信息TLV。在这里，口令是“jeff”，并使用十六进制的ASCII码字符来表示。

（5）支持的协议TLV

如图10-20所示，支持的协议TLV是由RFC 1195规定的，它用来指定PDU数据包的始发路由器所支持的协议——仅支持CLNP、仅支持IP或同时支持这两种协议。从那个时候起，TLV也用来对IPv6的支持。IPv4 NLPID是204（0xCC），而IPv6 NLPID是142（0x8E）。对于每一种所支持的协议，在TLV字段里都会有一个相应的1个八位组字节的网络层协议标识符（Network Layer Protocol Identifier, NLPID），这个标识符是由ISO/TR 9577规定的。IP协议的网络层协议标识符是0x81。在图10-15中的“Variable Length Field #2”部分显示的就是一个支持的协议TLV。

img490_1




图10-18 图10-12中显示的Hello PDU数据包末尾的可变长字段就是填充TLV，它把图10-12中显示的PDU大小增大到1497个八位组字节。加上3

个八位组字节的LLC头部和18个八位组字节的以太网头部，整个帧的大小就是最大为1518个八位组字节的以太网MTU的大小

img490_2




图10-19 认证信息TLV的格式

img490_3




图10-20 支持的协议TLV格式

（6） IP接口地址TLV

如图10-21所示，IP接口地址TLV是指发出PDU数据包的接口地址或IP地址。因为长度字段是1个八位组字节，因而IS-IS路由器的接口理论上最多可以有63个IP地址。在图10-17中的“Variable Length Field #4”部分显示的就是一个IP接口地址TLV，表明所捕获的Hello PDU数据包是由地址为10.1.3.1的接口发出的。

img491_1

图10-21 IP接口地址TLV的格式

3. IS-IS协议链路状态PDU格式

IS-IS协议中LSP的功能在本质上和OSPF协议中LSA的功能是一样的。一台L1路由器把L1类型的LSP泛洪到整个区域，用来确定它的邻接关系和这些邻接关系的状态。一台L2路由器把L2类型的LSP泛洪到整个第2层的域，用来确定它与其他L2路由器的邻接关系，并标识出通告L2路由器能够到达的地址前缀。

如图10-22所示，显示了一个IS-IS LSP的格式。这个格式对于L1 LSP和L2 LSP都是相同的。

img491_2

图10-22 IS-IS LSP的格式

- **PDU长度** ——是指整个PDU数据包的长度，用八位组字节数表示。
- **剩余生存时间（Remaining Lifetime）** ——在确认一个LSP过期之前等待的秒数。
- **LSP ID** ——可以是系统ID、伪节点ID或LSP数据包的LSP编号。LSP ID在“更新过程”小节中有更为详细的描述。
- **序列号** ——是一个32位的无符号整数。

- 校验和 ——对LSP内容的校验和。
- **P**位 ——是指分段区域的修复位。虽然这一位存在于L1和L2的LSP数据包中，但是它实际上只和L2的LSP数据包有关。当该位被设置为1时，表明始发路由器支持自动地修复区域的分段情况。Cisco IOS软件并不支持这项功能，因此Cisco路由器始发的LSP数据包中，该位始终设置为0。
- 区域关联位（**ATT**） ——是一个4位的字段，用来指明始发路由器是与一个或多个其他区域相连的。虽然区域关联位也存在于L1和L2的LSP数据包中，但是它们实际上只和L1/L2路由器始发的L1 LSP数据包有关。这4位用来表明相连的区域究竟使用哪一种度量方式。从左到右这4位依次表示：

- ☐ 位7：差错度量（Error）；
- ☐ 位6：代价度量（Expense）；
- ☐ 位5：时延度量（Delay）；
- ☐ 位4：缺省度量（Default）。

Cisco IOS软件仅支持缺省度量，因此位5～位7始终设置为0。

- 过载位（**OL**） ——链路状态数据库的过载位。在一般情况下，该位设置为0。收到过载位设置为1的LSP数据包的路由器，仍然会转发数据包到与这台始发过载信号的路由器直连的网络上，但是，不会再使用这台始发路由器作为转发数据包的透传路由器（transit router）了。
- 中间系统类型（**IS Type**） ——是一个两位的字段，用来指明始发路由器是L1路由器还是L2路由器：

- ☐ 00=未用；
- ☐ 01=L1；
- ☐ 10=未用；

☐ 11=L2。

一台L1/L2路由器可以根据收到的LSP是L1类型的LSP，还是L2类型的LSP来确定设置这两位值。

在L1的LSP数据包中可以使用下面的TLV字段：

- 区域地址（类型1）；
- 中间系统邻居（类型2）；
- 终端系统邻居（类型3）；
- 认证信息（类型10）；
- LSP缓存大小（类型14）；
- 扩展的IS可达性（类型22）；
- IP内部可达性信息（类型128）；
- 支持的协议（类型129）；
- IP外部可达性信息（类型130）；
- 流量工程路由器ID（类型134）；
- 扩展的IP可达性（类型135）；
- 动态主机名（类型137）；
- MT中间系统（类型222）；
- 多拓扑（类型229）；
- IPv6接口地址（类型232）；
- MT可达的IPv4前缀（类型235）；
- IPv6的IP可达性（类型236）；

- MT可达的IPv6前缀（类型237）；
- 实验用（类型250）。

在L2的LSP数据包中可以使用下面的TLV字段：

- 区域地址（类型1）；
- 中间系统邻居（类型2）；
- 区域分段指定的第2层中间系统（类型4）；
- 前缀邻居（类型5）；
- 认证信息（类型10）；
- LSP缓存大小（类型14）；
- 扩展的IS可达性（类型22）；
- IP内部可达性信息（类型128）；
- IP外部可达性信息（类型130）；
- 域间路由选择协议信息（类型131）；
- 支持的协议（类型129）；
- IP接口地址（类型132）；
- 流量工程路由器ID（类型134）；
- 扩展的IP可达性（类型135）；
- 动态主机名（类型137）；
- MT中间系统（类型222）；
- 多拓扑（类型229）；

- IPv6接口地址（类型232）；
- MT可达的IPv4前缀（类型235）；
- IPv6的IP可达性（类型236）；
- MT可达的IPv6前缀（类型237）；
- 实验用（类型250）。

如图10-23所示，显示了L1/L2路由器始发的一条L1类型的LSP。

img493_1




图10-23 协议分析仪捕获到的LSP信息

（1）中间系统邻居TLV（LSP）

中间系统邻居TLV用来在LSP中列出始发路由器的IS-IS邻居（包括伪节点），如图10-24所示。同时，它也列出了到达每一台邻居路由器的链

路的度量。

img494_1

图10-24 LSP中的中间系统邻居TLV的格式

- 虚拟标记（**Virtual Flag**）——这个字段虽然有8位长，但取值只有0x01或者0x00。当这个字段设置为0x01时，表示该链路是一个用来修复分段区域的第2层类型的虚链路。这时该字段只和支持区域分段能力的L2路由器相关，由于Cisco路由器并不支持这一特性，因此在Cisco路由器始发的LSP中该字段始终设置为0x00。
- **R**位——一是一个保留位，始终设置为0。

- **I/E位** ——该位和每个度量有关，用来指明相关的度量是内部度量还是外部度量。该位在中间系统邻居TLV字段中没有意义，因为对IS-IS域来说所有的邻居路由器都被定义为内部的。因此，在中间系统邻居TLV字段中该位始终设置为0。
- **缺省度量** ——是一个6位的缺省度量，用来表示始发路由器到达所列出的邻居的链路度量，大小范围在0~63之间。
- **S位** ——该位和每一个可选度量有关，用来指明相关的度量是被支持（0）的，还是不被支持（1）的。由于Cisco路由器不支持其他所有的3种可选度量，因此这一位总是被设置为1，并把相关的长度为6位的度量字段全部设置为0。
- **邻居ID** ——是指邻居的系统ID，再加上一个额外的八位组字节。如果该邻居是一台路由器，末尾的那个八位组字节就设置为0x00。如果该邻居是一个伪节点，那么系统ID就是指定路由器（DR），末尾的那个八位组字节就是伪节点的ID。

图10-25中显示了一个中间系统邻居TLV的部分信息。

（2）IP内部可达性信息TLV

如图10-26所示，IP内部可达性信息TLV列出了与通告该LSP的、路由器直连的路由选择域内的IP地址和相关的掩码信息。这个TLV使用在L1和L2类型的LSP中，但是从来不会出现在伪节点的LSP中。度量字段的含义是和中间系统邻居TLV中的度量字段一样的，但是它没有与可选度量相关联的I/E位。作为替代，这一位作为保留位并总是设置为0。像中间系统邻居TLV一样，这个TLV字段中的I/E位也总是设置为0，因为在这个TLV字段中通告的地址总是内部地址。如图10-27所示，显示了协议分析仪捕获到的一个IP内部可达性信息TLV的信息。

img495_1




图10-25 一个LSP中的中间系统邻居TLV的部分信息

img495_2

图10-26 IP内部可达性信息TLV的格式

（3）IP外部可达性信息TLV

IP外部可达性信息TLV列出了到达IS-IS路由选择域外部的IP地址和相关的掩码，这些外部的目的地址可以通过始发路由器的某个接口到达。IP外部可达性信息TLV的格式和图10-26中显示的IP内部可达性信息TLV的格式是相同的，但有一个例外，就是它的类型代码是130。其中I/E位用来确定所有4种度量的类型——I/E=0表示内部度量，而I/E=1表示外部度量。

img496_1

图10-27 协议分析仪捕获到的一个IP内部可达性信息TLV的信息

（4）域间路由选择协议信息TLV

如图10-28所示，域间路由选择协议信息TLV允许L2类型的LSP利用IS-IS域，来透传来自外部路由选择协议的信息。TLV字段具有一个相同的用途，就是提供给RIPv2、EIGRP和OSPF协议数据包的路由标记

（Route Tag）字段使用。路由标记将在第14章中介绍。

- 域间信息类型（**Inter-Domain Information Type**）——指定了在可变长度的外部信息字段中包含的信息类型。如果该类型字段设置为0x01，那么外部信息就会使用本地域间路由选择协议的方式。第14章包含了一个使用路由映射来设置这种本地信息的例子。如果该类型字段设置为0x02，那么外部信息就是一个16位的自主系统号，用来标记所有后续的外部IP可达性条目，直到LSP的结尾或者下一个域间路由选择协议信息TLV的出现。

img496_2

图10-28 域间路由选择协议信息TLV的格式

4. IS-IS协议序列号PDU报文

SNP通过描述数据库中的部分或者全部LSP的信息，对IS-IS链路状态数据库进行维护。在某些实例中，它们也用于请求LSP，以及隐性或显性地确认接收到的LSP。因此，SNP具有和OSPF中链路状态请求、数据库描述和链路状态确认消息的功能。

如图10-29所示，一台指定路由器（DR）将会周期性地以组播方式发送完全序列号数据包（CSNP），来描述伪节点数据库中的所有LSP信息。由于存在L1类型和L2类型的数据库，因此完全序列号数据包也可能是L1类型或者L2类型的。有时，某些链路状态数据库的信息量太大，以至于LSP的信息无法使用单个完全序列号数据包来描述。基于这个原因，完全序列号包头最后两个字段作为起始LSP ID（Start LSP ID）字段和结束LSP ID（End LSP ID）字段，一起用来说明完全序列号数据包中描述的LSP的范围。如图10-30所示，图中显示了这两个字段是怎样使用的。在这个完全序列号数据包中将会描述完整的数据库信息，因此，LSP ID的范围将开始于0000.0000.0000.00.00，并结束于ffff.ffff.ffff.ff.ff。如果需要两个完全序列号数据包来描述这个数据库，那么第一个完全序列号数据包的范围可以是0000.0000.0000.00.00～0000.0c0a.1234.00.00，而第二个完全序列号数据包的范围可以是

0000.0c0a.1234.00.01～ffff.ffff.ffff.ff.ff。

img497_1



图10-29 IS-IS协议完全序列号数据包的格式

img497_2




图10-30 协议分析仪捕获的信息显示了一个L1类型的完全序列号数据包头部信息

如图10-31所示，一个部分序列号数据包（PSNP）除了像前面描述的只是携带部分LSP的信息，而不是整个数据库的信息外，其他与完全序列号数据包相似。因此，不必像完全序列号数据包那样需要起始和结束字段。一台路由器可以在一个点到点的子网上发送部分序列号数据包来确认收到的LSP。在一个广播型的子网上，部分序列号数据包将会用来请求丢失的或者最新的LSP。和完全序列号数据包一样，部分序列号数据包也存在L1类型和L2类型。

img498_1

图10-31 IS-IS PSNP的格式

在SNP中，不论是CSNP还是PSNP，也不管是L1类型还是L2类型，它们都只用到了4个TLV字段：

- LSP条目（类型9）；
- 认证信息（类型10）；
- 可选的校验和（类型12）；
- 实验用（类型250）。

LSP条目TLV

如图10-32所示，LSP条目TLV总结了一个LSP中列出的该LSP的剩余生存时间、LSP ID、序列号和校验和。这些字段信息不仅可以确定某个LSP，而且可以完全地确定某个LSP的一个具体实例。如图10-33所示，显示了一个LSP条目TLV的部分信息。

img498_2

图10-32 LSP条目TLV的格式

img499_1

图10-33 CSNP的LSP条目TLV的部分信息，该CSNP如图10-30所示

10.1.5 IS-IS的扩展属性

正如读者在表10-4中以及相关的讨论中所看到的，在IS-IS协议扩展用来支持IPv4时已经增加了许多扩展属性，这些扩展属性有的为了支持可选的性能，有的改善了IS—IS协议的基本机制。在本小节我们将探讨这些扩展特性中最重要的一些内容。

有关IS-IS协议基本机制方面的一个好的处理操作是，它和OSPFv2协议的LSA不同，IS-IS协议碰到一个无法识别的TLV时，它只会忽略该TLV。这一特性使IS-IS协议在处理扩展性和向后兼容性方面变得更加容易。假如读者需要迁移你的网络以便支持一些新的特性，或者只是希望

有的路由器支持某个扩展属性而其他的路由器则不需要支持，那么担心对邻接关系的影响会比在OSPFv2协议中更少。在第9章中读者已看到，OSPFv3至少采用了该属性的部分特性，这使OSPFv3协议比OSPFv2协议具有更好的扩展性和广泛性。

1. 三方握手

在邻居之间建立邻接关系之前，邻居必须确保它们之间形成了双向通信。确保该过程的形成称作握手。对于双向握手来说，仅仅简单地发送和接收Hello数据包是不够的。因为你的邻居可能并未接收你所发出的Hello数据包。因此，对于你的邻居是否收到你所发出的Hello数据包作一些明确的确认是必要的，这就是三方握手（three-way handshaking）。OSPF协议总是使用三方握手，如果从邻居收到了Hello数据包，本地路由器就会在它发送的Hello数据包中列出所有这样的邻居。因此，如果一台OSPF路由器看到它的RID在所收到的Hello数据包的邻居字段中列出，那么它就可以知道这个Hello数据包的始发路由器已经收到了这台路由器的Hello数据包。这时，双向通信状态就可以得到确认。

在广播型的网络中，IS-IS协议也是使用三方握手机制的。如果从邻居收到了Hello数据包，本地路由器所发送的LAN Hello数据包就会在它的IS邻居TLV列出所有这样的邻居。而TLV也是在OSPF Hello数据包里列出邻居来达到同样的目的：IS-IS路由器在它收到的LAN Hello数据包的IS邻居TLV中看到它自己的SysID，那么它就会认为双向通信状态已经确认建立了。

但是，正如已经所提及的那样，点到点Hello数据包是不携带IS邻居TLV的。ISO 10589 规定通过点到点链路只能建立双向握手，并且要求点到点的传输介质应该是可靠的。但是在现实当中，点到点链路可能经常是不可靠的；当然，我们也不会仅仅因为在网络中可能存在一条不能满足某些含糊的可靠性要求的链路，就不把IS-IS协议作为一个可能的IGP协议选择。

为了修正这个问题，RFC 3373指定了一个点到点三方邻接TLV（Point-to-Point Three-Way Adjacency TLV）。如图10-34所示，它是一个类型为240的TLV，列出了发起它的路由器所知道的所有邻居的SysID，并且它也指出了始发路由器在该链路上可能的邻接关系状态：正常、初始化，

或失效（Up、Initializing，或Down）。

img500_1

图10-34 点到点三方邻接TLV的格式

2. 跨域范围（**Domain-Wide**）的前缀分发

正如读者在本章前面的部分了解到的，L1区域的缺省特性和OSPF的完全末梢区域非常相似。换句话说，从区域L2到L1不通告任何前缀；相反的，L1/L2路由器设置ATT位，并且L1路由器加载一条到达最近的L1/L2路由器的缺省路由。事实上，RFC 1195规定，禁止从L2区域向L1

区域通告前缀。

但是，这样的规定也不总是可接受的。如果存在多台L1/L2路由器，有时我们更希望选择到达目的地最近的那台L1/L2路由器，而不是选择所在区域离出口最近的那台L1/L2路由器。为了达到这个目的，L1路由器必须拥有相应前缀的必要信息和它们在本地区域外的路径代价；这也就意味着必须把这个前缀从L2区域通告到L1区域。在IS-IS协议的专业术语中，这种情况称为“路由泄漏（route leaking）”。

在这里可能存在潜在的困难（这也是RFC 1195中禁止像这样“向下的”路由泄漏的原因），如果在L1区域内具有多台L1/L2路由器——这很可能是读者起初为什么从L2区域向L1区域泄漏路由的原因，这可能会引起潜在的区域间路由选择环路问题。如图10-35所示，图中演示了这个问题的存在。L1/L2路由器Hague通过一些L2 LSP学习到前缀192.168.1.0/24的信息。如果需要将该前缀信息通告到它的L1区域内，那么它必须使用一个IP内部可达性TLV在它的L1 LSP中通告这个前缀信息。但这个L1 TLV在L1区域内扩散时，区域内的另一台L1/L2路由器Rotterdam将会收到它。然而，由于这条前缀信息是在L1 LSP中扩散的，路由器Rotterdam没有办法得知它是始发于L1区域外部的。因此，这台路由器将会通过一条L2 LSP将这条前缀信息通告回L2区域。这样，在L2路由器上就存在路由选择环路了：L2路由器认为到达前缀192.168.1.0/24的路径存在于L1区域内。

img501_1

图10-35 根据基本的RFC 1195规则，从L2区域向L1区域泄漏前缀信息会存在该前缀被通告回L2区域的风险，这样会产生一个路由选择环路

OSPF协议并不存在这种问题，因为在OSPF协议中，对于非骨干区域外

部的前缀都是通过类型3的LSA通告的。在一个非骨干区域内的其他ABR路由器将不会把从类型3的LSA学到的前缀信息通告到区域0中。

在RFC 2666中，通过在IP内部可达性TLV和IP外部可达性TLV中定义一个称为“Up/Down (U/D)”的位，来为该问题提供了一种解决方案。在图10-26中显示的格式中，查看IP内部可达性TLV的格式（并请记得IP外部可达性TLV的位格式也是相同的），请注意这个八位组包含了一个I/E位和一个6位的缺省度量字段，它们开始于一个保留位。在图10-36中，这个保留位就变成了U/D位。在指定了类型字段后，就可以为IP内部可达性TLV和IP外部可达性TLV定义该一位了。

当一台L1/L2路由器从L2区域向L1区域通告一条路由时，它将会设置这个U/D位。任何其他L1/L2路由器收到在L1 LSP里设置了U/D位的前缀，都不会在L2 LSP中通告这个前缀。如果有一台L1/L2路由器不能识别这个U/D位，那么它将忽略这一位。因此，如果读者正在使用L2到L1的路由泄漏设计，那么确保你所有的L1/L2路由器都能识别这个扩展特性是很重要的。

在本章稍后的10.2.8小节中，将会示范怎样设置L2到L1的路由泄漏技术。

3. 扩展度量（Wide Metrics）

流量工程（Traffic Engineering, TE）是一个主要与多协议标签交换（Multiprotocol Label Switching, MPLS）网络相关联的功能，在MPLS网络中的数据包的一些子集能够依赖用户指定的约束条件以不同的方式进行转发。换句话说，不像IGP协议那样总是选择单一的最短路径通过一个网络，业务流（traffic flows）能够在不同的路径上传播。这种做法既可以帮助网络提高整个带宽的利用率，也能够提供区分服务——例如，确保时延敏感的流使用最短的路径，而其他的业务流使用较长的路径。

流量工程的一个关键是需要使用比度量更为详细的接口参数进行通信；使用用于共享路径设计的IGP协议和用于共享这些TE接口参数的接口信息就变得有意义了。OSPF与IS-IS协议都被扩展以便能够携带TE接口参数；对于IS-IS协议来说，使用以下两种新的TLV：

- 扩展的IS可达性（类型22）；

- 扩展的IP可达性（类型135）。

img502_1

图10-36 缺省度量字段的第八位，原来是保留的，现在被重新定义为Up/Down位

在RFC 3784中，详细描述了这些TLV以及它们所支持的TE参数。但是，MPLS流量工程的讲解已经超出了本书的讲述范围。然而，我们关注这两个新的TLV是因为它们提供了一种新的重要性能，并且能够用在流量工程以外的地方。

正如本章前面所提及的，在IS-IS最初的格式里，有关IS-IS协议的一个问题是：仅仅6位的度量字段不能满足一个大型网络所需要的足够的度量尺度。这两个新的TLV则可以支持一个更大的度量尺度，因此称为“扩展度量（wide metrics）”。扩展的IS可达性TLV使用一个24位的度量字段，而扩展的IP可达性TLV使用一个32位的度量字段。

扩展的IS可达性TLV的格式如图10-37所示。当启用扩展度量时，这个TLV就用来在LSP中替代类型2的IS邻居TLV。对于我们来说所关注的是类型2的TLV，它列出了进行SPF计算的邻居以及它们的度量，注意是24位的度量字段。子TLV（sub-TLV）字段是用于TE的参数信息，和这里的讨论无关。但是，这种TLV是允许嵌套的TLV，就是说，是子TLV（sub-TLV），或嵌入其他TLV内的TLV，注意到这一点是有用的。这样的TLV将会为开发人员带来更多的灵活性。

扩展的IP可达性TLV的格式如图10-38所示。当启用扩展度量时，这个TLV就用来替代IP内部可达性信息和IP外部可达性信息TLV。因此，这个TLV可以出现在L1和L2的LSP中。在图示中，读者可以看到通过这个TLV通告的前缀具有一个32位大小的度量值。对于从L2到L1区域的路由泄漏也具有一个Up/Down标记位。而在扩展的IS可达性TLV中，子TLV仅仅与流量工程相关联（S位表示子TLV的当前状态）。

img503_1

图10-37 扩展的IS可达性TLV

img503_2

图10-38 扩展的IP可达性TLV

在IOS软件系统中启动扩展度量的命令是**metric-style wide**。在本章后面的10.2.6小节和10.2.7小节将会讲述使用扩展度量的例子。

4. IPv6的IS-IS路由选择

在第9章，读者已经看到OSPF协议需要一个全新的版本来支持IPv6。而另一方面，IS-IS协议通过两个新的TLV就可以很容易地进行扩展来支持IPv6了。在撰写本书时，IS-IS协议的IPv6扩展属性依然是在Internet草案（Internet-Draft）阶段，但是很快它就可能形成一个RFC——可能就在

读者正在阅读本章的时候。

IS-IS协议通过在其支持的协议TLV中包含IPv6 NLPID 142（0x8E）来支持IPv6。这两个支持IPv6的TLV是：

- IPv6可达性（类型236）；
- IPv6接口地址（类型232）。

如图10-39所示，IPv6可达性TLV可以说是为了实现IPv4中IP内部可达性信息与IP外部可达性信息TLV相同的功能的。但是，同样相似的功能与类型135的扩展IP可达性TLV更加相近，这是因为以下两个原因：

- 这个TLV是用于同时通告内部和外部的缀信息的；
- 这个TLV包括了一个32位的度量字段，因此支持扩展度量特性。

img504_1

图10-39 IPv6可达性TLV

从图中可以看到，对于每一个前缀都有一个32位的度量字段和用于从L2向L1路由泄漏的Up/Down位。X位表示这个前缀是由内部始发的（X=0），还是由外部始发的（X=1）。S位表示是否存在子TLV。

如图10-40所示，图中显示了IPv6接口地址TLV，它的功能等价于IPv4中类型为132的IP接口地址TLV：它通告了始发这个TLV的接口的地址信息。并且和IPv4中相对应的，它能够同时被Hello数据包和LSP数据包携带。当TLV出现在Hello数据包中时，它所通告的地址就是始发接口的链

路本地地址。如果TLV出现在LSP数据包中，那么它所通告的地址就不是始发接口的链路本地地址，而是地区或全球范围的地址。

img505_1

图10-40 IPv6接口地址TLV

即使IP接口地址TLV能够携带最多63个32位的IPv4地址，但IPv6地址是这个长度的4倍，因此，这就限制了IPv6的接口地址TLV最多可以携带15个IPv6地址。

在本章随后10.2.6小节中讲述了一个有关IPv6的IS-IS的配置案例。

5. 动态主机名交换

使用IS-IS协议工作存在一个操作上的困难，就是它很难将多台不同路由

器显示的SysID与正确的路由器相关联。标识IPv4地址是不容易的，但是我们往往为了方便而给它取一个长一点的比较熟悉的名字。对于SysID，就像示例10-1中显示的那样确实存在实际的挑战和困难。

在RFC 2763中提供了一个动态主机名TLV（类型137）的说明，它提供了一个解决上述困难的简单方案。这个简单的TLV在它的数值字段提供了最多255位的空间用来携带一个路由器名字的ASCII码。通常情况下，支持这个扩展的实现方式是，将所配置的路由器主机名插入到这个字段中去；然后就可以在任何非伪LSP中携带这个TLV。如示例10-8所示，当需要显示一台路由器时，动态主机名TLV中的ASCII码值就会用来替代它的SysID，以便更加容易的进行识别。

示例10-8 这个示例重新显示了示例10-1中IS-IS邻居表的内容，可以看出动态主机名交换机制的好处

img505_2

6. 多拓扑结构

现代IP网络通常会在一个通用的网络基础设施上提供多种网络服务，一般通过基本的网络服务模块如IPv4、IPv6和多播等实现。使用一个通用的网络基础设施提供所有的网络服务，而不愿为每一种服务组建各自独立的网络，这会在投资成本上带来巨大的好处：大大降低了投资开支，较少的设备维护，简化的操作流程，以及为数不多的操作人员。但是，

各个基本的服务模块需要不同的路由选择拓扑结构。也许IPv4是无处不在的运行，但是IPv6应该只是局限在IPv4域的某个部分。多播服务可能也需要局限于IPv4域的某些子域，但是却与IPv6具有不同的拓扑结构。

多拓扑（Multi Topology, MT）路由选择允许读者在一个通用的网络基础设施上组建这些路由选择的网络子集。各个不同的拓扑可以使用多拓扑ID（MT ID）来标识。当前定义的MT ID列在了表10-5中。这些MT ID是设计在路由器接口上的，用来表示这个接口属于什么拓扑；每一个接口都可以具有一个或多个MT ID。邻接关系不是具体到特定的MT的，而仍然是在所有的IS-IS邻居之间建立的。但是，这些LSP都会被标记上相应的MT ID，并且每一个拓扑结构都会运行各自的SPF计算。在每一台路由器上，都会为每一个拓扑维护各自的路由表，而且路由表中的每一个路由条目都是基于各自的SPF计算而产生的。

表10-5 用于**MT IS-IS**的多拓扑ID

MT ID	拓 扑
0	标准（缺省）拓扑（IPv4单播路由选择拓扑）
1	IPv4 带内管理（in-band management）
2	IPv6单播路由选择拓扑
3	IPv4多播路由选择拓扑
4	IPv6多播路由选择拓扑
5~3995	IETF统一保留
3996~4095	保留，用于开发、试验以及专有特性

IS-IS支持的MT路由选择目前还处于Internet草案阶段。在这个草案中指定了几个支持MT的TLV:

- 多拓扑中间系统（类型222）；
- 多拓扑（类型229）；
- 多拓扑可达的IPv4前缀（类型235）；
- 多拓扑可达的IPv6前缀（类型237）。

当一台MT IS-IS路由器发送Hello数据包时，它会包含一个或多个多拓扑

TLV，用于始发接口所属于的每一个拓扑，如图10-41所示。如果一台邻居路由器没有包含Hello数据包中的所有的多拓扑TLV，那么这个邻居就会被认为仅仅属于缺省的IPv4拓扑（MT ID为0）。在一个点到点的链路上，如果两个邻居之间没有任何共同的MT ID，那么在这两个邻居之间将不会形成邻接关系。但是，这和一个广播型链路上的情况是不同的，在广播型链路上即使邻居之间没有任何共同的MT ID，邻居之间也会形成一个邻接关系。这是因为指定路由器的选举是独立于MT IS-IS扩展特性的，一台不支持扩展特性的路由器依然能够被选举为指定路由器（DR），因此在相同的层次的所有路由器一定是邻接的。

多拓扑TLV不仅可以由Hello数据包携带，也可以由LSP数据包携带。正如图10-41所显示的，这个TLV可以分别为每一个拓扑表示过载（使用O位）和L2关联（使用A位）。

img507_1

图10-41 多拓扑TLV的格式

多拓扑中间系统TLV的格式如图10-42所示，它和前面讲到的扩展IS可达性TLV几乎完全相同，除了对于每一个始发接口所属的每种拓扑都要有

各自单独的多拓扑中间系统TLV外，它和扩展的IS可达性TLV一样，也服务于同样的目的。

img507_2

图10-42 多拓扑中间系统TLV的格式

多拓扑可达的IPv4前缀TLV的格式如图10-43所示，多拓扑可达的IPv6前缀TLV的格式如图10-44所示。多拓扑可达的IPv4前缀TLV和多拓扑可达的IPv6前缀TLV，在功能上分别与扩展的IP可达性TLV和IPv6可达性TLV相同。它们同样通告前缀，但同每一个给定的拓扑相关联。

在本章后面的10.2.7小节中讨论了一个配置多拓扑模式IS-IS的示例。

img508_1

图10-43 多拓扑可达的IPv6前缀TLV的格式

img508_2

图10-44 多拓扑可达的IPv6前缀TLV的格式

7. Mesh组

虽然有利于SONET和MPLS的技术正在快速消退，但是帧中继和ATM网络却依然经常作为许多大型网络的核心传输介质。帧中继和ATM网络经常使用的拓扑结构如图10-45显示的那样，是用于提供连接性的全网状连接的虚电路。但是，过多的网状连接或全网状连接的网络容易受到

过重的泛洪扩散负载影响。全网状连接是指所有的路由器都与其他所有的路由器相连接。

由于每一台路由器和其他任何一台路由器都进行连接，那么，在一台路由器扩散一条LSP时，其他路由器会立即收到该LSP，如图10-46所示。这是进行泛洪扩散所需要的。

img509_1

图10-45 承载IP网络的ATM和帧中继网络基础设施经常配置成每一个节点都和其他所有节点进行连接，也就是说这些节点之间是全网状连接的

img509_2

图10-46 在一个全网状连接的网络里，来自某台路由器的传播会立即被其他所有路由器接收到

现在存在的问题是，接收路由器和其余的路由器之间都存在链路连接，而它们并没有办法知道所扩散的LSP已经被其他路由器收到。因此，根据泛洪扩散的基本水平分割规则，这些路由器将会把LSP从没有收到这个LSP的其他每个接口扩散出去，如图10-47所示。这个例子显示，如果某个网络上具有 n 台路由器，那么网络中的每一台路由器都要扩散 $n-2$ 条不必要的LSP：除了它本身和收到这条LSP的路由器，LSP将会被扩散到其余的每一台路由器。根据二次方程式，可以计算出扩散了 $(n-1)(n-2)$ 或 $n^2 - 3n + 2$ 条不必要的LSP。 [\[20\]](#)

img510_1

图10-47 即使每一台路由器已经收到了扩散的LSP，它们仍然会根据泛洪扩散的规则，来发送该LSP到它的每一个邻居，除了扩散该LSP的那个邻居

我们在这里检查一下上面网络的情况，它额外的扩散负载并不是太重：对于6台路由器，产生了 $n^2 - 3n + 2 = 20$ 条不必要扩散的LSP。但是，假定一个网络存在100台全网状连接的路由器，那么就会产生 $n^2 - 3n + 2 = 9702$ 条不必要扩散的LSP。而且这还仅仅是来自一台路由器刷新它的LSP。考虑到每一台路由器都会刷新它们的LSP，同时每台路由器可能有多条LSP，另外除了刷新计时器触发的LSP扩散外还有其他一些活动，我们可以看出每一次泛洪扩散所产生的不必要的LSP是多么巨大。当然，在现代网络中，这个通信量与整个网络要处理的所有数据包的通信量相比也许不是不可接受的，但是对于许多网络架构设计师来说，任何无效率的设计都违背了审美的情趣。

对于这些为了简单地寻找更加简洁设计的工程师来说，在RFC 2973中提供了有关IS-IS协议的*mesh* 组（*mesh groups*）。Mesh组的机制允许读者在清楚地确认路由器的邻居已经收到某个LSP时可以避免再扩散这个

LSP。Mesh组将一个接口定义为下面3种模式之一：

- 不活动模式（Inactive）；
- 阻塞模式（Blocked）；
- 设置模式（Set）。

不活动模式简单的来说就是虽然路由器支持mesh组，但是没有mesh组在接口上启用，LSP是正常扩散的。

但一个接口处于阻塞模式时，它就不再扩散任何LSP。如图10-48所示，图中显示了怎样通过阻塞接口来减少图10-45中网络的扩散负载。在这里，图10-45中全网状连接的拓扑已经简化成一个“环形”扩散拓扑；当然，正常的数据包转发依然能够使用全网状连接的网络结构。在这个扩散拓扑中，每一台路由器都有两个邻居而不再是 $n-1$ 个邻居了。这样虽说还有一点不必要的扩散，但是这已经大大减少了图10-47中所显示的情况。

对于图10-48中部分接口被阻塞的拓扑也是需要权衡的，或者说总是需要权衡的。首先，当每一台路由器只和其他两台路由器具有非阻塞的扩散连接时，如果这两条连接都失效了，那么，即使这台路由器还有其他完好的连接到达邻居，扩散也将会被破坏。其次，就是对收敛时间的影响。图10-46中全网状连接的扩散拓扑，可以确保每一台路由器在要扩散的LSP一经发出时就可以被收到；但是在图10-48中，部分接口扩散被阻塞的拓扑意味着，当某些情况下要扩散的LSP需要通过一台或几台路由器传输后才会被所有的路由器接收到。

img511_1

图10-48 在一些接口上阻塞扩散可以减少整个扩散的负载

设置模式与阻塞模式相比，给我们提供了更多的灵活性，在仍然降低扩散负载的情况下（虽然没有阻塞模式降低的那么多），提供了在阻塞模式中存在的冗余性降低与收敛时间增加的一种折衷方案。在设置模式下，可以定义带有编号的mesh组，并分配相应的接口到对应的组。在收到一条LSP时，这条LSP就在属于某个编号mesh组的接口上被收到。然后，除了那些与接收到这条LSP的接口同属于一个组的接口外，这条LSP会在其他所有的接口上扩散出去。

在图10-49中，显示了将图10-45中的拓扑配置成了两个mesh组：组1和组2。假设一台路由器正在发起一条LSP，它把这条LSP扩散到每一个接口上而不管这个接口属于哪个组；这种最开始的扩散看起来比较像图

10-46中显示的情形。一些邻居在属于组1的接口上收到这条LSP，而一些邻居则在属于组2的接口上收到这条LSP。

接着，接收LSP的路由器将这条LSP从与接收该LSP的接口所属的组不同的mesh组的接口上扩散出去。如果LSP是在属于组1的接口上收到的，那么它只能从属于组2的接口上扩散出去，反之亦然，如图10-50所示。所有这些扩散的LSP当然是不必要的，每一台路由器在始发路由器最开始扩散该LSP时就收到了它的一份拷贝。与图10-48中的扩散模型相比，可以看出虽然这些不必要的扩散比阻塞模式加重了一些负载，但是它仍然比图10-47中显示的负载小许多。而且，由于最开始的扩散到达所有的路由器，因此收敛时间没有受到影响。

不活动模式、阻塞模式和设置模式可以混和使用来组成一个复杂的拓扑，以便获取读者所希望的扩散模型。在实际使用中，阻塞模式比设置模式用的更为普遍。但是，如果读者真的选择使用mesh组，就应该意识到在所有的情况下，都是以牺牲全网状连接的扩散拓扑所具有的健壮性为代价来降低泛洪扩散的负载的。

在IOS软件系统中，读者可以在路由器的接口上使用命令**isis mesh-group blocked** 来阻塞扩散，或者使用命令**isis mesh-group number** 来把某个接口分配到已编号的mesh组中。

img512_1

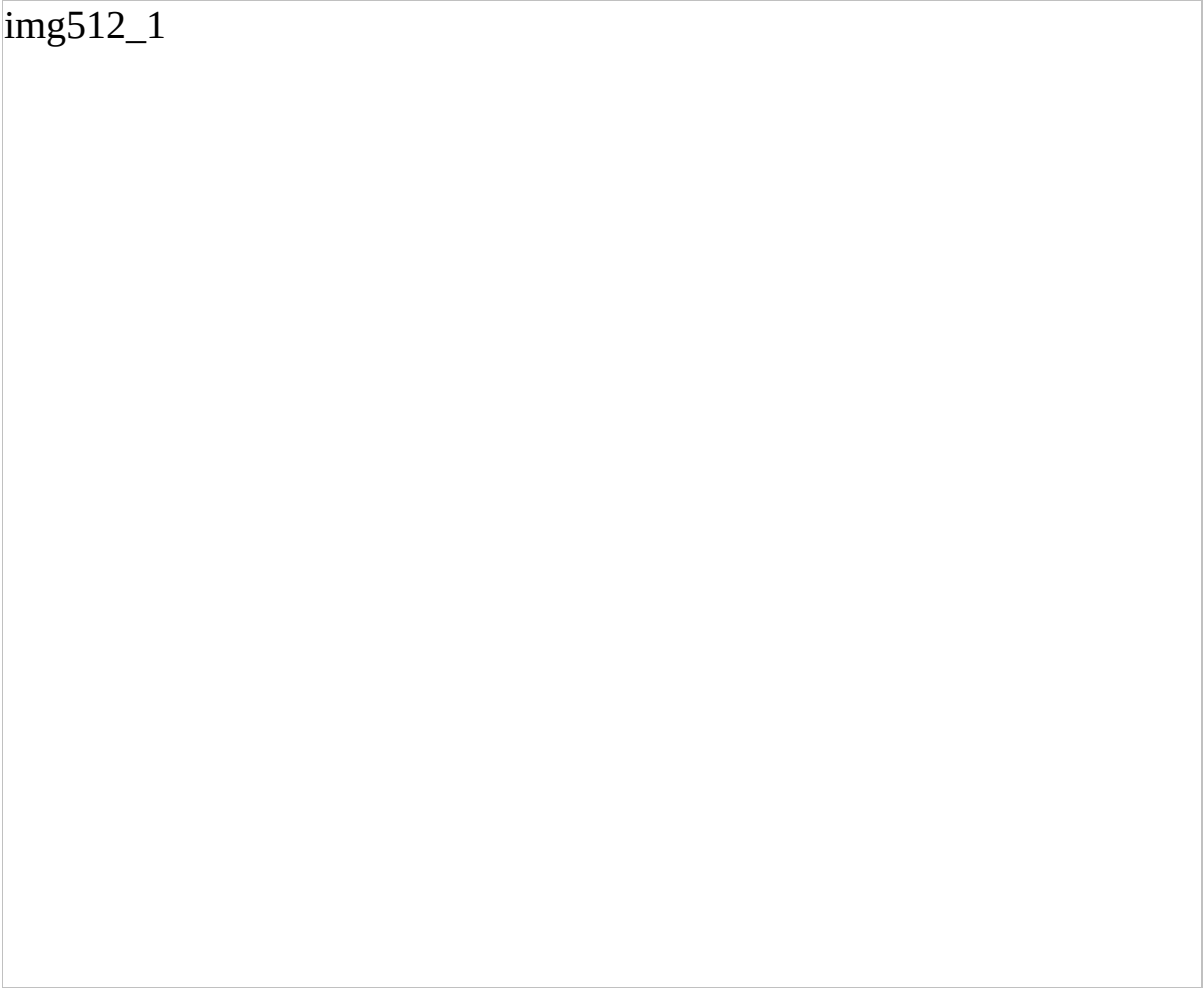


图10-49 设置模式允许我们创建已编号的mesh组，并将接口分配到这些组内

img512_2

图10-50 路由器将收到的LSP扩散到所有与接收该LSP的接口所属的组不同的mesh组的接口上

10.1.6 泛洪扩散的时延

虽然mesh组能够在比较密集的网状网络扩散过程中控制发送大量不必要的LSP，但是，如果一个网络不稳定的话泛洪扩散也会成为一个问题。例如，一个快速频繁波动的链路会引起大量的LSP扩散，每一次都会触发所有区域内的路由器进行SPF计算，并会耗尽网络资源。

IOS软件系统提供了几种在这些情况下减少扩散影响的工具。第一个方法就是在路由器碰到不稳定的情况下减少本地始发的LSP的扩散，例如存在忽好忽坏波动的链路。这种情况下，在扩散一个LSP之前强制加上一个时延就可能避免每一次波动都引起一次扩散的情况发生。例如，假设有一条链路每20s波动一次，而扩散每一分钟进行一次，这样就只会在每第5个波动才会引起一次扩散。

但是，我们并不希望总是延迟扩散：在网络处于稳定的情况下，等待一分钟才进行新的LSP的扩散，这会使收敛时间大大增加。因此，在IOS软件中利用一个指数补偿算法，在开始生成一个LSP之前增加一个非常

小的时延，只有50ms。如果在前一个LSP生成后的5s内由于某个事件产生了另一个LSP，那么该路由器将会等待5s钟再生成新的LSP。这个时延一直等到网络稳定10s后才取消，而重新恢复50ms的LSP生成间隔。

这个指数补偿算法的缺省状态可以通过命令**lsp-gen-interval** 来改变，以下是该命令的可选项：

- **lsp-initial-wait** ——是路由器产生一个LSP正常的等待时间。缺省值为50ms，变化范围是1~120000ms。
- **lsp-second-wait** ——是指路由器在产生第一个和第二个LSP之间所需要的时延。缺省值为5000ms（即5s），它的变化范围是1~120000ms。在第二个LSP生成后，前面的时延就会加倍，并作为每一个后续LSP生成的时延。因此，如果第二个LSP的生成等待了5s，那么在第二个和第三个LSP的生成之间就会等待10s，在第三个和第四个LSP的生成之间等待20s，在第四个和第五个LSP的生成之间等待40s，依此类推，一直到lsp-max-wait所设定的时延。这个参数也给出了为什么叫指数补偿算法这个名字的原因。
- **lsp-max-wait** ——是指LSP生成之间所允许的最长时延。如果在网络发生连续不稳定的情况时，为了使时延不会增加到影响生成LSP的长度，很有必要设置这个参数。这个参数的缺省值是5s，它的变化范围在1~120s之间。

虽然在局部出现不稳定时可以采用**lsp-gen-interval** 抑制LSP的生成，但在某台特定的路由器出现性能问题并因为大量的扩散产生过载时也可能会产生问题。惟一的解决办法是升级你的路由器。但是在过渡阶段可以利用一些可选项来控制泛洪扩散到邻居。第一个可选项是**isis lsp-interval** 命令选项，它可以用来在一个接口上（因此也是对特定的邻居）改变LSP发送之间的缺省时延。缺省时延是33ms。例如，可以将时延设置为100ms，那么每0.1s只能传送1个LSP，或每秒只能传送10个LSP，而不能更快。

对于不能满足性能要求的那些邻居，LSP重传也会产生问题。如果一台路由器正在吃力地处理接收到的LSP，那么它可能会延迟确认这个LSP，从而引起它的邻居重传这个LSP。如果这台路由器从一开始就比较吃力，那么重传可能会使情况更加恶化。重传一条LSP的缺省等待时间是5s。通过增加等待时间可以稍微保护一下性能不足的邻居；使用命

令**isis retransmit-interval**可以增加等待时间，最大为65535s。

即使你使用命令**isis retransmit-interval**增加了重传一条LSP的等待时间，可能仍然会存在问题，等待时间可能会使多条LSP超时，而它们可能都应该被重传。读者可以使用命令**isis retransmit-throttle-interval**来调整传输这些需要重传的LSP之间的等待时间。这样我们就可以利用命令**isis retransmit-interval**增加等待重传确认的时间，而使用命令**isis retransmit-throttle-interval**在等待时间超时的情况下，增加每一个重传的LSP之间的时间间隔。

虽然这些命令给了我们一些控制泛洪扩散的手段，但它们应该仅仅在极端的情况下使用。在绝大多数的情况下，缺省的参数值不应该改变；在看来有必要修补这些参数值的时候，读者应该找出改变的原因。通常情况下，进行路由器升级或提高链路的可靠性才是更好的解决方案。

10.1.7 提高SPF的效率

IOS软件系统使用以下两个机制来提高SPF算法的效率：

- 递增SPF（Incremental SPF，iSPF）；
- 部分路由计算（Partial Route Calculations，PRC）。

当将一台末梢路由器增加到网络中时——也就是说这台路由器仅通过一条链路增加到网络上——那么区域内的所有路由器都不需要重新计算SPF。相反，只需要增加这台路由器到SPF树就足够了。如果还没有加载到SPF树上的链路在某些方面的变化不会影响SPF树，那么就根本没有必要为此而运行SPF计算。iSPF考虑到这些情况，只根据拓扑改变的程度大小而运行SPF计算。iSPF也能够限制SPF计算的范围。也就是说，如果一个变化只影响拓扑的有限部分，那么iSPF就可以把SPF的重新计算限制在拓扑变化的范围内。

另一个没有必要触发进行SPF计算的变化是IP前缀的增加、删除或度量变化。检测到这样一台变化的路由器会通过IP可达性TLV（或功能上等价的TLV）扩散一个LSP，从而通告这个变化。但是这个LSP不需要触发进行SPF计算，这就是部分路由计算：首先检测所收到的LSP，然后确认如果不是需要进行SPF计算的拓扑变化，例如一个新的IP可达性TLV或新的IS邻居TLV，那么就不运行SPF计算。

在网络不稳定的时期，在SPF计算的运行之间加入更长的时延可能跨越多条LSP的接收时间。也就是说，如果很多LSP正在进行泛洪扩散，两个SPF计算的运行之间的等待可能意味着并不是每次收到一条LSP而运行SPF计算，而是意味着路由器可能收到了很多LSP，并为所有这些LSP运行一次SPF计算。IOS软件系统对于运行SPF计算也采用了类似抑制LSP生成的指数补偿算法，抑制LSP生成的指数补偿算法已经在前面章节中讲述过了。命令**spf-interval**应用于完整的SPF运行，而**prc-interval**则应用于部分路由计算。在这两种情况中，像抑制LSP的生成一样，读者可以指定一个初始化等待时间、为每一次后续运行间隔加倍的第二等待时间，以及不能超出的最大等待时间。SPF指数补偿算法缺省的初始化等待时间是5500ms，第二等待时间的缺省值是5500ms，而最大等待时间的缺省值是10s；对于PRC来说，它们的缺省值分别是2000ms、5000ms和5s。但是，作为扩散的时延，改变SPF时延的缺省值通常不是一个好主意。解决频繁地进行SPF运行的实际可行的方法是，可靠的链路和网络部件，并且尽可能的使用区域来减少泛洪扩散的范围。

10.2 集成IS-IS协议的配置

集成IS-IS协议在本书介绍的IP路由选择协议中显得比较独特，这有两方面的原因。首先，IS-IS协议是惟一个必须作为一个进程启动又要在单独的接口上启动的协议。其次，IS-IS协议是惟一的一个开始并不是为IP协议设计的IP路由选择协议。由于集成IS-IS协议使用CLNS PDU数据包而不是IP数据包，因此IS-IS协议的配置就不如其他路由选择协议的配置那么清楚直观了。

10.2.1 案例研究：IPv4集成IS-IS的基本配置

在一台Cisco路由器上配置一个集成IS-IS，需要以下4个步骤：

步骤1： 确定路由器所在的区域和启动IS-IS协议的接口。

步骤2： 使用router isis命令来启动一个IS-IS进程。 [\[21\]](#)

步骤3： 使用net 命令来配置NET地址。

步骤4： 使用命令ip router isis在相应的接口上启动集成IS-IS。这个命令不仅在转发接口（和IS-IS邻居相连的接口）上必须增加，而且在一个和末梢网络相连的接口也必须要配置，这里的末梢网络是指需要IS-IS协议来通告的IP地址。

如图10-51所示，显示了一个包括6台路由器的小型网络，它被分成了两个区域。使用NET地址表示方式，区域1和区域2将分别表示为00.0001和00.0002，而它们各自的系统ID是每台路由器E0或FastEthernet0/0接口的MAC地址标识符。表10-6中显示了使用该信息进行编码得到的NET地址。

img515_1




图10-51 区域1表示成NET地址方式是00.0001，区域2表示成NET地址方式是00.0002。每一个NET的系统ID都是接口E0或FastEthernet0/0的MAC地址标识符

img516_1


表10-6
址

图10-51中路由器IS-IS配置用到的NET地址

在路由器Paris、London、Brussels和Amsterdam上的配置显示在示例10-9～示例10-12中。

示例**10-9** 路由器**Paris**的配置

img516_2



示例**10-10** 路由器**London**的配置

img516_3

示例**10-11** 路由器**Brussels**的配置

img516_4

示例**10-12** 路由器**Amsterdam**的配置

img517_1

路由器Berlin和Rome的配置基本相似。在这里有一个需要注意的配置细节是，在Amsterdam路由器的配置中启动了CLNS协议的路由选择功能。CLNS路由选择对于IS-IS协议处理CLNS PDU是必需的。当我们在创建一个IS-IS进程的时候，就自动地启动了CLNS路由选择。在IOS软件的某些版本中，我们可能在配置中看到命令**clns routing**，虽然这在配置中并不是要求输入的。

参见示例10-13，示例中显示了路由器Paris的路由表。请注意，这里路由表中同时包含了L1路由和L2路由。在缺省条件下，Cisco路由器默认是L1/L2路由器。这个事实也可以通过查看路由器的IS邻居表清楚地看出来（参见示例10-14）。

示例10-13 路由器Paris的路由表中同时显示了L1路由和L2路由，表明这台路由器是一台L1/L2路由器

img517_2

示例**10-14** 路由器**Berlin**的IS邻居表显示出路由器**Paris**和**Rome**都是**L1/L21**路由器

img517_3

这里请注意，路由器Berlin的IS邻居表中列出了每一个邻居的名字。主机名是使用“动态主机名交换”小节中讲述的类型137的TLV动态交换的。为了显示与每个主机名相关联的系统标识符，可以使用命令**show isis hostname**，参见示例10-15所示。

示例10-15 使用命令**show isis hostname**可以显示与已知主机名相关联的系统ID

img518_1

由于在图10-51的网络中每一台路由器都是L1/L2类型的，因此，每一台路由器都会同时形成L1类型的邻接关系和L2类型的邻接关系。也正因为如此，每一台路由器也将会同时维护一个L1类型的链路状态数据库和一个L2类型的链路状态数据库。L1区域和L2区域完全重叠了。例如，示例10-16中显示的路由器Amsterdam的链路状态数据库；其中，L1类型

的数据库中包含了一条始发于路由器Amsterdam^[22]的LSP数据包和一条始发于路由器Brussels的LSP数据包。同时，它还包含了一条始发于路由器Brussels的伪节点LSP数据包（Brussels.02-00），用来描述路由器Brussels和Amsterdam之间的以太网链路。请记住，作为伪节点LSP数据包的LSP ID是可以辨别出来的，因为伪节点LSP数据包的LSP ID的倒数第二个八位组字节（也就是伪节点ID）是非零的。

示例10-16 路由器Amsterdam同时具有一个L1类型的链路状态数据库和一个L2类型的链路状态数据库，这表明该路由器是一台L1/L2路由器

img518_2

这3条L1的LSP表明路由器Amsterdam除了它本身外，所知道的惟一的L1路由器是Brussels。这个单一的节点是可以预料到的，因为在区域2中路由器Brussels是惟一另外的路由器。路由器Amsterdam的L2数据库显示Amsterdam和IS-IS域内的每一台路由器都形成了L2邻接，当然这也是可以预料到的，因为每一台路由器都是一台L1/L2类型的路由器。

10.2.2 案例研究：更改路由器的类型

在如图10-51这样的小型网络中，在路由器上保留它们缺省的IS-IS类型

是可以接受的。但是，当网络规模不断增大时，使用这些缺省的类型将越来越不能被接受。因为，这不仅要消耗大量的路由器CPU和内存去处理和维持两个链路状态数据库，而且要消耗大量的缓存和带宽去处理和泛洪每一台路由器始发的L1和L2类型的IS-IS PDU。

在图10-51中的路由器Paris、Berlin和Amsterdam可以配置成L1路由器，因为它们都没有和其他区域直连的链路。在路由器上可以使用命令is-type来更改缺省的路由器类型。例如，要把路由器Berlin变成一台L1类型的路由器，它的配置如示例10-17所示。


示例10-17 路由器Berlin配置为一台L1路由器

img519_1

路由器Paris和Amsterdam的配置也和上面类似。比较一下路由器Paris在示例10-18中的路由表和在示例10-13中的路由表，可以发现L2类型的路由已经被删除了。同样地，比较一下路由器Amsterdam在示例10-19和在示例10-16中的路由表，可以发现现在路由器Amsterdam只剩下L1数据库了。

示例10-18 当路由器Paris配置成一台L1路由器后，它的路由表中就只包含到达它本身所在区域内的目的地址的路由了

img519_2



示例**10-19** 当路由器**Amsterdam**配置成一台**L1**路由器后，它就只包含**L1**的链路状态数据库了

img519_3

回忆一下，在前面曾经讲述过LSP中的区域关联位（ATT位），一台L1/L2路由器会通过设置ATT位来告知L1路由器它具有区域间的连接。示例10-20中显示了路由器London的LSP和路由器Rome的LSP都将ATT位设置为1了，即ATT=1。因此，路由器Paris将会知道把区域间的通信量发送到路由器London或者Rome。换句话说，路由器Paris会有一条到达路由器London或Rome的缺省路由，而且到达路由器London的路径将成为优先路径，因为路由器Paris到达它的度量更小。在示例10-18中，并没有在路由器Paris的路由表中显示出这条缺省路由（0.0.0.0）。

示例10-20 路由器**London**和**Rome**始发的**L1 LSP**中设置了**ATT=1**，这表明它们具有到达其他区域的连接

img520_1

在IOS软件的旧版本中，IP进程不能直接解释ATT位。ATT位是一个CLNS的特性。当运行IOS软件的旧版本时，如果在L1路由器中没有自动地创建缺省路由，将有两种方法解决这个问题。第一种解决方法是在路由器的接口上除了启用IP协议的IS-IS，另外再启用CLNS协议的IS-IS。例如，修改路由器London和Paris的串行接口的配置，参见示例10-21和示例10-22。

示例10-21 在路由器**London**的接口上配置**CLNS**的**IS-IS**，启动**ATT**位处理

img520_2

示例10-22 在路由器**Paris**的接口上配置**CLNS**的**IS-IS**，启动**ATT**位处理

img520_3

第一种解决方法需要IS-IS协议运行在一个CLNP/IP的混和环境里，但是如果IS-IS协议仅仅是作为一个单一的IP路由选择协议使用的话，那么启用CLNS路由选择仅仅为了生成一条IP缺省路由就显得十分没必要了。而第二种解决缺省路由问题的方法是在L1/L2路由器上配置一条静态路由，并且使用命令**default-information originate** 配置IS-IS协议来通告这条缺省路由。在图10-51的区域2中使用这个方法，路由器Brussels的配置参见示例10-23所示。

示例**10-23** 路由器**Brussels**配置成发起一条缺省路由

img520_4

关于缺省路由和**default-information originate** 命令更为详细的介绍将在第12章中讲述。

10.2.3 案例研究：区域的迁移

在OSPF协议中如果更改区域的地址，就必须考虑和预计好网络中断的时间。但是，在IS-IS协议的设计中，能够在网络不中断的情况下允许更改区域地址。正如在10.1节中讲述的，Cisco路由器可以最多配置254个L1区域地址。为了使两台路由器能够形成一个L1邻接，它们必须至少具有一个公用的区域地址。在允许具有多个区域地址的情况下，新的邻接关系能够在旧的邻接关系中断时替代它。这种方法在某些情况下会显

得非常有用。例如，在合并区域或拆分区的时候、在为一个区域重新编号的时候，或者在同一个IS-IS域内同时运行多个编址机构分配的区域地址的时候，等等。

举个例子，在图10-52（a）中的路由器都具有一个区域地址01（这些路由器当中的任何一台设备的NET地址看上去应该像010000.0c12.3456.00一样）。在图10-52（b）中，这些路由器另外分配了一个区域地址03。虽然实际上并没有形成多个邻接关系，但是这些路由器还是可以识别出它们具有多个公共的区域地址。在图10-52（c）中，区域01已经从某一台路由器上移走了。这3台路由器仍然保留着邻接关系，因为它们至少还有一个公共的区域地址。最后，在图10-52（d）中，区域地址01从这3台路由器上全部移走了，这时，这3台路由器全在区域03中。可以看出，在区域迁移期间并没有什么时候丢失邻接关系。

img521_1

图10-52 每台路由器都支持多个区域地址，可以使区域的更改变得比较容易

假定因为某些原因，认为图10-51中的网络正在使用的区域地址安排不合法，而被强令要求必须符合GOSIP。这时通过注册U.S.GSA机构后，可以使用下面的资源来组成新的NET 地址：

AFI: 47

IDI: 0005

DFI: 80

AAI: 00ab7c

Reserved: 0000

RDI: ffe9

Areas: 0001 (area 1) , 0002 (area 2)

根据上述信息，表10-7中显示了新的NET地址。

表10-7 图10-51中的路由器分配到了新的GOSIP
格式的NET地址

路由器	NET地址
Paris	47.0005.80.00ab7c.0000.ffe9.0001.0004.cl50.e700.00
Berlin	47.0005.80.00ab7c.0000.ffe9.0001.0010.7b3c.6bd3.00
London	47.0005.80.00ab7c.0000.ffe9.0001.00b0.6430.lde0.00
Rome	47.0005.80.00ab7c.0000.ffe9.0001.0004.cl50.flc0.00
Brussels	47.0005.80.00ab7c.0000.ffe9.0002.0005.5e6b.50a0.00
Amsterdam	47.0005.80.00ab7c.0000.ffe9.0002.0000.0c8d.34fl.00

更改区域地址的第一步是增加新的NET地址到路由器上，但是不改变原来的NET地址。路由器Rome上的IS-IS配置参见示例10-24。


示例10-24 在转换期间路由器Rome配置了两个NET

img522_1

其他5台路由器的配置也和上面类似。可以使用带关键字**detail** 的命令 **show isis database** 来查看结果（参见示例10-25所示），或者也可以使用命令**show clns is-neighbors** 来查看结果（参见示例10-26所示）。在这两个数据库中，可以看出网络内的每一台路由器都是和多个区域相连的。

示例**10-25** 在路由器**Rome**的链路状态数据库中，**LSP**显示出图**10-51**中网络的所有路由器都正在通告两个区域地址

img522_2



img523_1

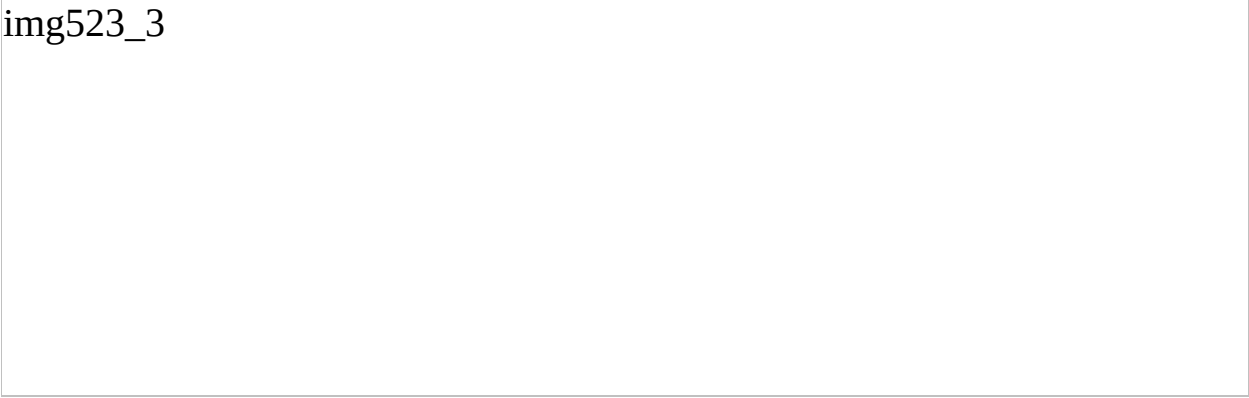
示例**10-26** 路由器**Rome**的**IS-IS**邻居表中也显示出每台邻居路由器都与多个地址相关联

img523_2

区域迁移的最后一步是从所有的路由器上删除原来的**NET**地址语句。例如，在路由器**Rome**的**IS-IS**配置中输入命令**no net 00.0001.0004.c150.flc0.00**。在示例10-27中，显示了在路由器**Rome**上删除了以前的**NET**语句后，该路由器数据库中的一些**LSP**信息。

示例**10-27** 路由器**Rome**的数据库中的**LSP**显示出只有一个区域地址

img523_3



img524_1

10.2.4 案例研究：路由汇总

在第8章中，已经介绍了在链路状态协议的区域之间如何进行路由汇总。关于路由汇总更完整的讲述将会在第12章讲述的缺省路由部分介绍。这里先简要地描述一下汇总路由的好处：

- 汇总路由可以有效地减小LSP的大小，这样也就减小了链路状态数据库的大小，从而也节省了路由器的CPU和内存消耗。
- 汇总路由可以隐藏掉区域内部网络的不稳定影响。如果仅仅是一个汇总地址范围内的地址发生了改变或一条链路的状态发生变化，那么并不会通告到做汇总的区域外部。

当然，汇总路由也有以下一些缺点：

- 汇总路由的效果依赖于能够进行汇总的连续的IP地址范围，因此地址分配必须进行仔细规划。
- 汇总路由由于隐藏区域内的细节因而减少了路由的精确性。如果具有多条进入汇总区域的路径，那么将无法确定最佳的路径。

路由汇总可以在IS-IS的配置下使用命令**summary-address**来启动。配置了这条语句后，任何在汇总地址范围内的更具体的目的地址都将被抑制，而汇总路由的度量会选择它所有更具体的地址中更小的度量。

在图10-53中，显示了包含3个区域的一个IS-IS网络。在这里，区域1内的地址可以汇总为172.16.0.0/21，而区域3内的地址可以汇总为172.16.16.0/21。路由器Zurich、Madrid和Bonn的配置参见示例10-28～示例10-30。[\[23\]](#)

示例10-28 路由器**Zurich**的配置执行路由汇总

img525_1

示例**10-29** 路由器**Madrid**的配置没有执行路由汇总

img525_2

示例**10-30** 路由器**Bonn**的配置执行路由汇总

img525_3

img525_4

图10-53 路由器Zurich和Bonn将区域1和区域3汇总到区域2

这里注意，路由器Madrid由于没有L1类型的邻居，因而配置成一台L2路由器。路由器Zurich和Bonn正在汇总它们各自的区域到L2类型的骨干。参见示例10-31所示，在路由器Madrid的路由表中显示出了路由汇总的结果。

示例**10-31** 路由器**Madrid**的路由表显示出路由器**Bonn**和**Zurich**通告的汇总地址

img525_5

img526_1

10.2.5 案例研究：认证

IS-IS协议的认证可以使用明文口令或HMAC-MD5。有两种方法配置明文口令。这两种认证方式对于防止来自网络的攻击提供了一个很弱的安全机制，但是对于防止因为配置出错或未受权的路由器造成网络服务的中断还是比较有效的。其中一种明文配置模式使用钥匙链，它可以在配置文件中被加密以防止别人通过查看配置文件无意获取口令。没有钥匙链的明文认证被认为是“老式”口令。

Cisco IOS软件支持3个级别的IS-IS认证：邻居之间、区域范围和IS-IS域范围。这3个级别的认证可以单独使用也可以一起使用。IS-IS认证的规则如下：

- 当在邻居之间进行认证时，互相连接的路由器接口必须配置相同的口令；

- 当在邻居之间进行认证时，必须分别为L1和L2类型的邻接关系配置各自的认证；
- 当在邻居之间进行认证时，可以使用明文或MD5认证；
- 当认证在整个区域范围内有效时，区域内的每一台路由器都必须使用相同的认证模式和具有共同的钥匙串；
- 当认证在整个IS-IS域范围内有效时，IS-IS域内的每一台L2和L1/L2类型的路由器都必须使用相同模式的认证，并使用共同的钥匙串。

在IS-IS中配置认证的步骤和在RIPv2与EIGRP中的配置步骤是一样的。这些步骤重复如下：

步骤1： 定义一个带有名字的钥匙链。

步骤2： 在钥匙链上定义一个或一组钥匙。

步骤3： 在某接口上，或者为L1（区域范围）或L2（域范围）的IS-IS实例启用认证，并指定使用的钥匙链。

步骤4： 为某接口，或者为L1或L2的IS-IS实例指定将使用明文或MD5认证。

步骤5： 可选地配置钥匙的管理。

在网络中配置认证可以不中断路由器之间的邻接关系。为了完成这项功能，命令**isis authentication send-only** 必须首先配置在需要使用认证的所有路由器上。这条命令在接口上用来进行邻居之间的认证，在ISIS路由选择进程下用来进行区域范围或域范围的认证。在认证完全配置好之后可以删除这条命令。

如果在邻居之间进行认证，先配置钥匙链，命令**isis authentication key-chain** 引用预先配置的钥匙链，接着使用命令**isis authentication mode** 在直连的接口上配置认证类型。L1 与L2邻接可能引用不同的钥匙链。相邻的路由器必须共享共同的钥匙串或口令。在一个接口上可以指定任意一个或同时指定两个层的钥匙链，并且每一层的钥匙串可以相同也可以不同。当配置完后，钥匙串可以由IS-IS邻居之间的L1或L2 Hello数据包

内的认证信息TLV携带。

举个例子，在图10-53中的路由器Geneva、Zurich和Madrid上配置了认证，这3台路由器的相关配置参见示例10-32～示例10-34。

示例**10-32** 路由器**Geneva**的认证配置

img527_1

示例**10-33** 路由器**Zurich**的认证配置

img527_2

示例**10-34** 路由器**Madrid**的认证配置

img527_3

因为路由器Geneva和Zurich之间的邻接关系是L1类型的，因此只指定了一个L1类型的钥匙链引用。路由器Zurich和Madrid之间只存在L2类型的邻接关系，因此只指定了一个L2类型的钥匙链引用。这里要注意，如果没有指定关键字**level-1** 或者**level-2**，那么命令**isis password**、**isis authentication mode** 以及**isis authentication key-chain** 将会默认为是L1和L2。

注意路由器Geneva的配置。路由器Geneva连接到路由器Zurich的接口配置了老式明文口令。为了建立邻接关系，这个口令必须和路由器Zurich上定义的钥匙串相同。 [\[24\]](#)

如果要在一个区域内进行认证，可以使用命令**isis authentication mode mode level-1** 和命令**isis authentication key-chain name level-1** 来定义认证模式，并在IS-IS配置下引用一个钥匙链。当该模式是**text** 时，这两条命令一起用来替代老式的命令**area-password**。在接口配置模式下使用命令**isis authentication** 指定的钥匙串是在Hello数据包中传送的，而在IS-IS进程下使用**level-1**认证命令指定的钥匙串是在所有的L1 LSP、CSNP和PSNP中传送的。因此，邻居级别的口令只可以用来验证邻接关系的建立，而区域级别的口令则可以用来验证L1类型的链路状态信息的交换。如果区域认证没有配置正确，路由器将仍然可以形成邻接关系，但是不会进行L1 LSP的交换。

在图10-53中的区域3上配置区域口令，路由器Bonn和Frankfurt的配置参

见示例10-35和示例10-36。

示例10-35 路由器Bonn的区域口令配置

img528_1

示例10-36 路由器Frankfurt的区域口令配置

img528_2

路由器Frankfurt的IOS软件不支持新式的认证，因此使用命令**area-password**。注意该口令和路由器Bonn上定义的钥匙串是相同的。

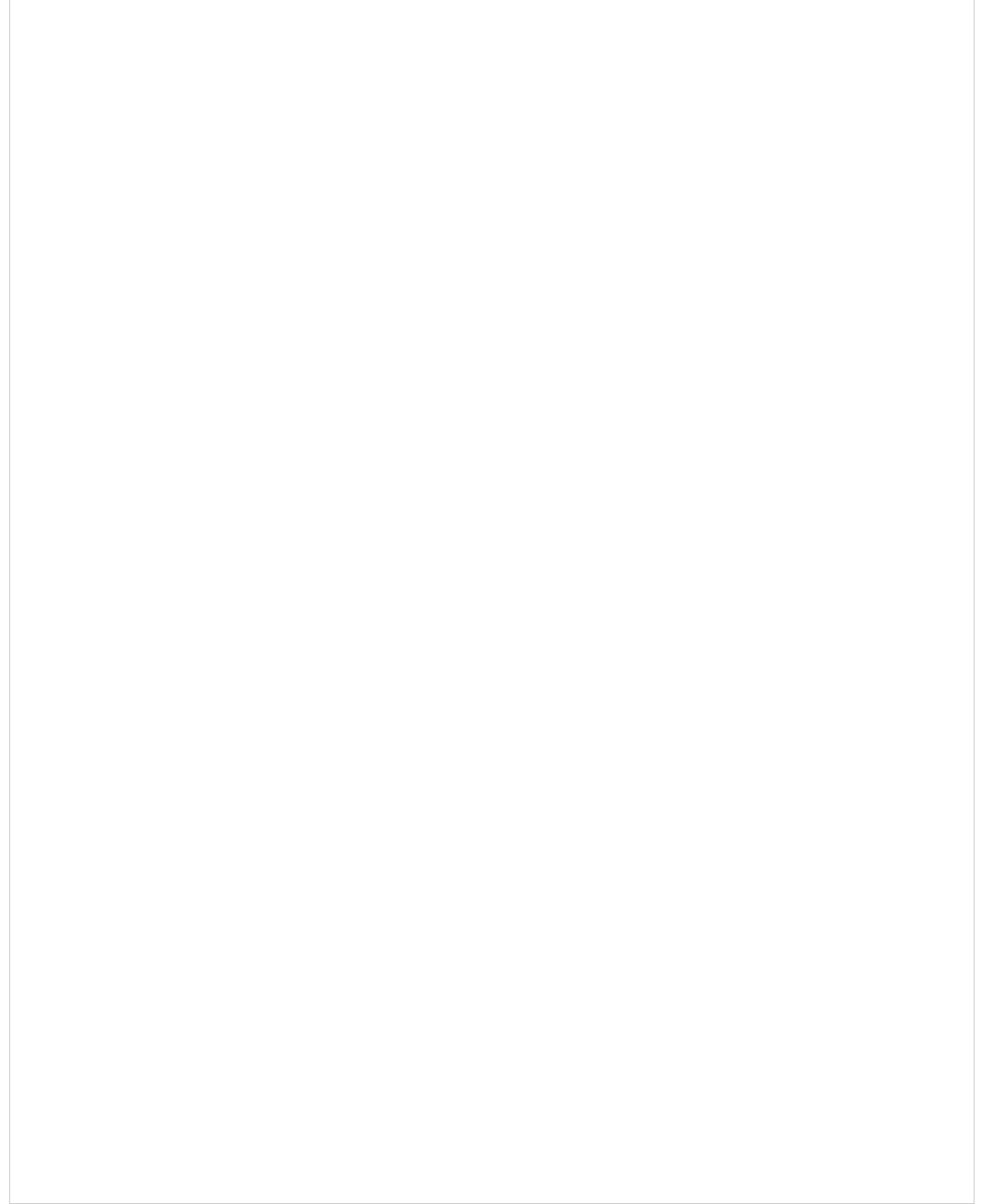
如果要在IS-IS域范围内配置认证信息，可以在IS-IS进程下使用带关键字**level-2**的命令**authentication key-chain**和**authentication mode**。这些配置命令可以和老式命令**domain-password**共同使用。但两种风格的命令不能同时配置在同一台路由器上。定义在钥匙链中被这些认证命令引用的钥匙串将在L2 LSP、CSNP和PSNP中传送。因而，IS-IS域认证可以用来验证L2路由信息的交换。像区域认证一样，IS-IS域认证不会去验证L2类型的邻接关系，但是会验证L2 LSP的交换。

在图10-53中的网络上配置IS-IS域认证，只需要在路由器Zurich、Madrid和Bonn上配置就可以了，因为路由器Geneva和Frankfurt是L1类型的。相关的配置参见示例10-37～示例10-39。

示例10-37 路由器Zurich的域认证配置

img528_3

img529_1



示例**10-38** 路由器**Madrid**的域认证配置

img529_2

示例**10-39** 路由器**Bonn**的域认证配置

img529_3

10.2.6 案例研究：IPv6集成IS-IS的基本配置

集成IS-IS对于IPv4和IPv6协议（还有CLNS）只需计算单个SPF来创建单个拓扑。如果在网络中配置了IPv4和IPv6，那么所有的接口和所有的路由器都必须配置这两种协议。

在路由器Geneva和Madrid之间增加一条链路到图10-53的网络中。路由器Geneva不再只是L1路由器了。IPv6也添加到该网络中。在图10-54中显示了新的地址配置。图中的路由器已经配置了集成IS-IS协议。为了运行IPv6协议下的IS-IS协议，可以先利用命令`ipv6 unicast-routing`在全局模式下启动IPv6路由选择，然后在接口上启动IPv6的IS-IS协议。在每一个具有IPv4地址的接口上都增加了一个IPv6地址和IPv6 IS-IS协议。路由器Geneva的新配置参见示例10-40。

img530_1

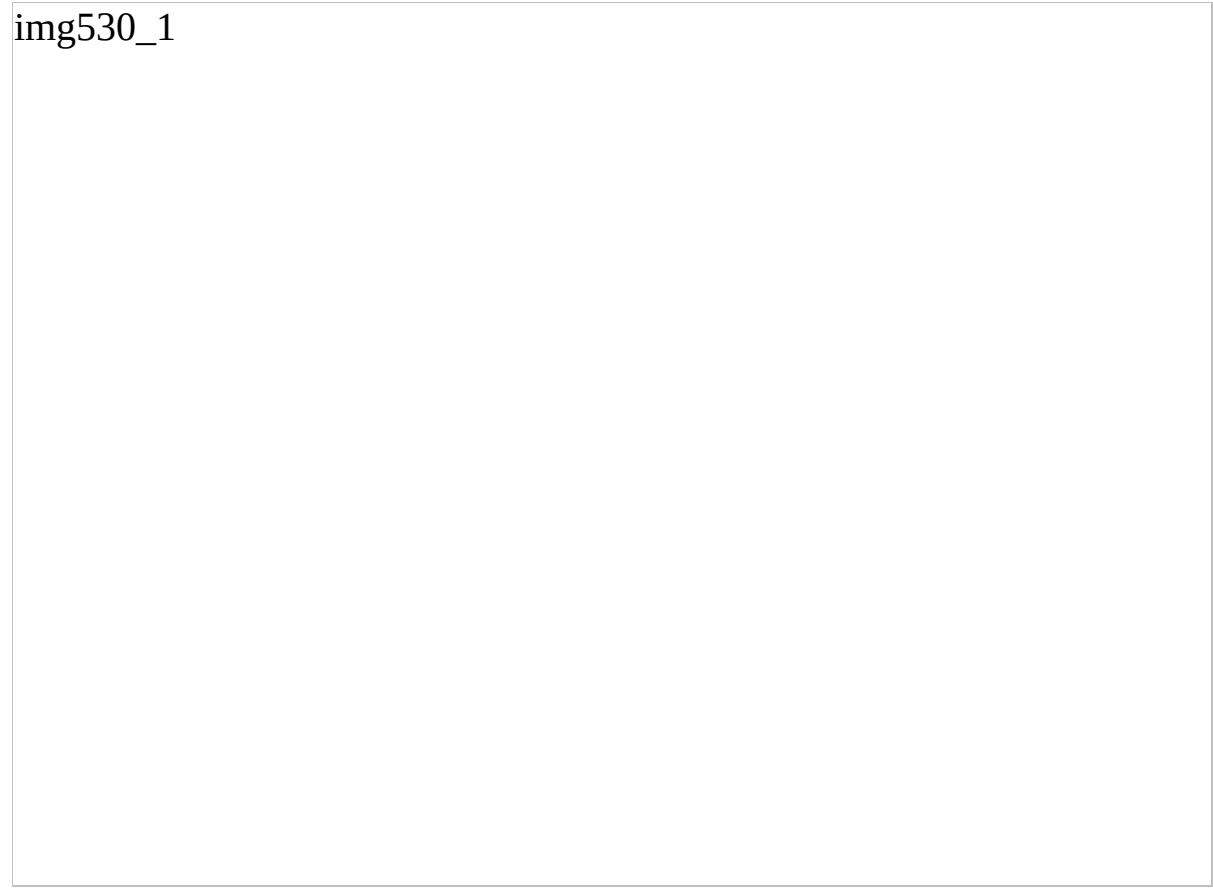


图10-54 在网络上增加了IPv6协议的IS-IS协议

示例**10-40** 路由器**Geneva**的**IPv6 IS-IS**配置

img530_2

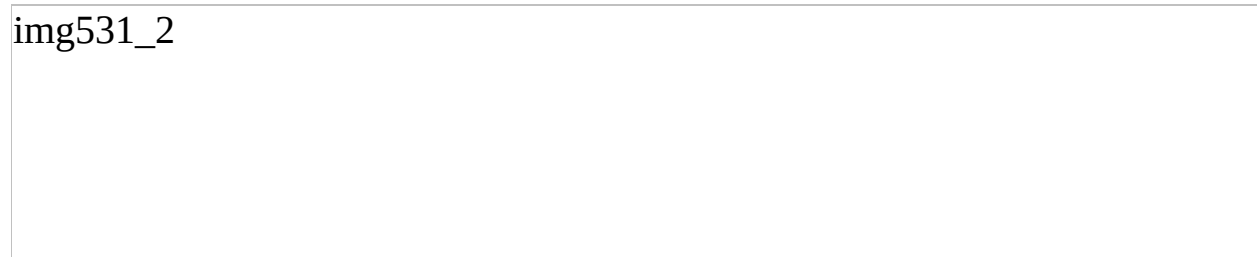


img531_1

IS-IS路由选择进程没有改变。IPv6地址已经添加到每一个接口上，并且在每一个接口上都运行了IPv6协议的IS-IS。其他路由器的配置也与上面类似。

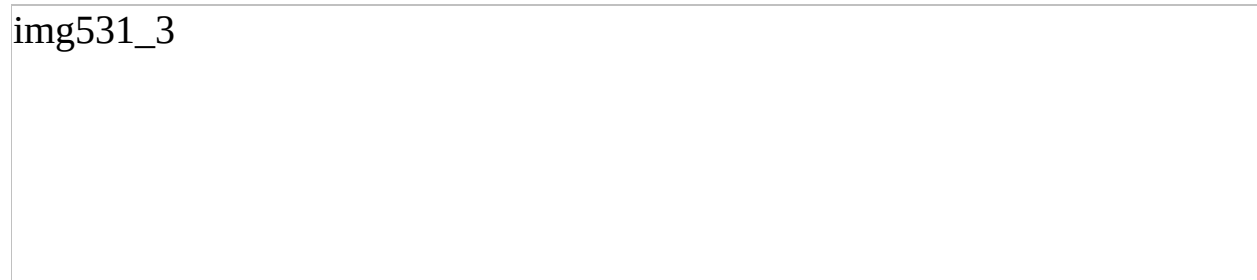
图10-54中的IPv6地址也可以类似IPv4地址那样进行汇总。路由器Zurich、Geneva和Bonn在将路由通告给L2路由器时进行了汇总。IPv6地址的汇总是在全局IS-IS路由选择进程模式下，通过指定IPv6地址簇和配置希望汇总的地址范围来实现的（参见示例10-41～示例10-43）。

示例**10-41** 路由器**Zurich**正在汇总**IPv6**前缀路由



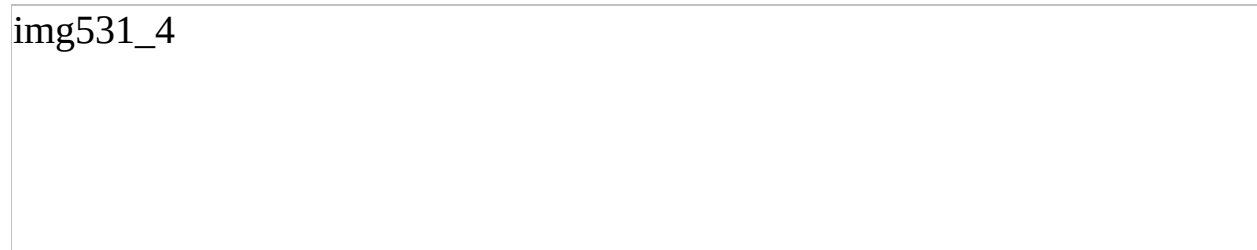
img531_2

示例**10-42** 路由器**Geneva**正在汇总**IPv6**前缀路由



img531_3

示例**10-43** 路由器**Bonn**正在汇总**IPv6**前缀路由



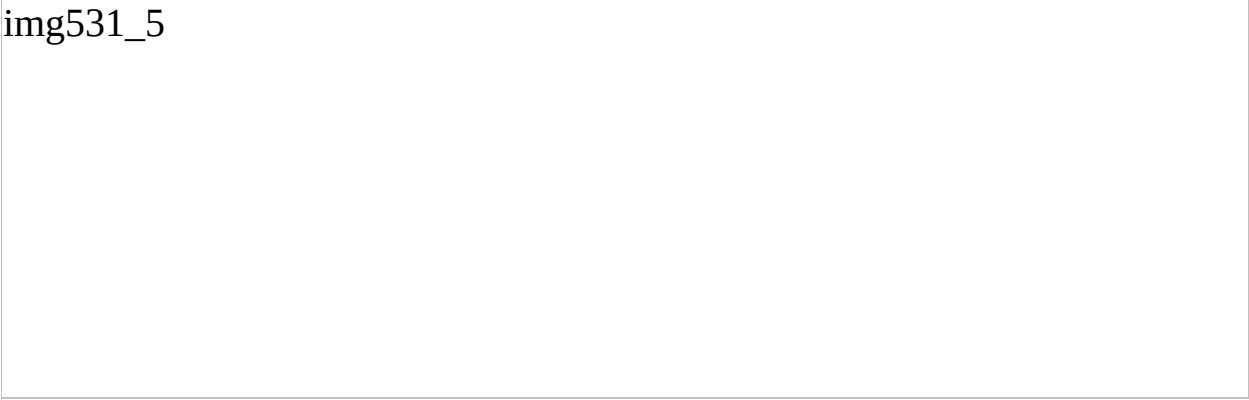
img531_4

在示例10-44中，显示了路由器Madrid的IPv6路由表，包含了路由器Zurich、Geneva

和Bonn汇总的地址范围。

示例**10-44** 路由器**Madrid**的**IPv6**路由表中显示了由路由器**Zurich**、**Geneva**和**Bonn**汇总的地址范围

img531_5



img532_1

IPv4和IPv6共享相同的拓扑结构，因此也共享同样的度量值。IPv6的TLV使用扩展的度量，但在缺省情况下，IPv4使用较窄幅度的度量，最大值为63。因此，IPv6的度量也受限为63。每一个接口的缺省值为10。度量的类型可以通过命令**metric-style** 来更改。关键字**wide**、**transition** 或**wide transition** 可以使路由器在网络重新配置期间发送和接收较窄和较宽幅度的度量。通过命令**show clns protocol** 的输出可以显示度量的类型，参见示例10-45所示。

示例**10-45** 路由器**Zurich**配置为缺省的较窄幅度的度量

img532_2

正如示例10-45中所看到的，路由器Zurich仅仅发送和接收较窄幅度的度量。这里请注意度量的类型对于IPv4和IPv6并不是专有的。

参见示例10-46所示，显示了路由器Zurich改变后的度量类型配置。

示例**10-46** 路由器**Zurich**的度量类型配置为**transition**模式，可以同时接收和发送宽幅与窄幅度的度量

img533_1

关键字**transition** 将会使该路由器发送和接收宽幅度的度量和窄幅度的度量。示例10-47显示了路由器Zurich的结果。

示例**10-47** 宽幅度的度量和窄幅度的度量都可以生成和接收

img533_2

10.2.7 案例研究：过渡到多拓扑模式

在图10-54中的网络出现了一个问题。路由器Frankfurt的IOS软件不支持IPv6协议。缺省情况下，当IPv6协议配置在路由器Bonn和网络中其余的路由器上后，IPv6协议也就添加到由IPv4协议为IS-IS的每一层创建的IS-IS拓扑中了。因为单一的拓扑是为IS-IS的每一层存在的，因此这个拓扑是由IPv4和IPv6协议共同共享的，IPv4和IPv6协议必须同时配置在每一条链路和每一台路由器上。如果使用单一的拓扑，并同时配置IPv4和

IPv6协议，那么具有IPv4地址的链路也都必须具有IPv6地址。这时，在路由器Bonn上就可以看出问题了。当配置IPv6协议时，它和路由器Frankfurt之间的邻接关系就无法创建。参见示例10-48，使用调试命令**debug isis adj-packets**显示了有关IPv6地址的问题所在。

示例**10-48** 使用调试命令**debug isis adj-packets**显示了单一拓扑模式下有关IPv6地址的一个错误

img533_3

IOS软件系统除了支持缺省的单一拓扑模式，还支持多拓扑模式的IS-IS。[\[25\]](#)在单一拓扑模式下，IPv4、IPv6和CLNS都共享相同的IS-IS拓扑。在配置了多拓扑模式后，IOS软件系统可以支持两种拓扑：一种与IPv6相关，另外一种与IPv4和CLNS相关。对于每一种拓扑都会有各自独立的SPF计算。

正如在“多拓扑结构”部分中讲述的，单拓扑的IS-IS和多拓扑的IS-IS使用不同的TLV交换信息。一台不能支持多拓扑模式的路由器将无法理解多拓扑TLV。一台多拓扑模式的路由器可以处理单拓扑模式的TLV：如果在邻接连接上没有收到多拓扑TLV，那么这台接收路由器就假定它的这个邻居是IPv4/CLNS拓扑的一部分。这在单拓扑模式和IPv6都已经配置在网络中，并且你希望把网络迁移到多拓扑模式的时候可能会出现问題。在迁移期间，IPv6网络的区域会变得不可到达。

IOS软件系统提供了一个过渡模式（transition mode），在这种模式下可以发送单拓扑和多拓扑的TLV，但是路由器只能在多拓扑模式下操作。在所有的路由器都配置完成后，将会删除过渡模式。

在配置多拓扑模式之前，在路由器上必须配置用在多拓扑TLV中的扩展度量。

在图10-54中显示的网络上配置了多拓扑模式。所有路由器最初的配置都是过渡模式，具体配置参见示例10-49。

示例10-49 图10-54中的所有路由器都具有以下配置

img534_1


路由器Frankfurt不支持IPv6和多拓扑模式，它惟一的配置变化就是将度量改变为宽幅度的度量类型。由于路由器Frankfurt不发送多拓扑TLV，所以Frankfurt路由器和它的链路都保留为缺省的IPv4拓扑模式。

从单拓扑模式到多拓扑模式的转换需要两条命令：第一条命令将度量类型从窄幅度改变为宽幅度，第二条命令启动多拓扑模式。新的拓扑共享NET和IPv4拓扑中的L1/L2边界。

在路由器Bonn上可以看到两个不同的拓扑。路由器Bonn上的IS-IS IPv6拓扑显示了从L1路由器到L2路由器的IPv6路径。这里有两条L1路由器的条目。一条是对应路由器Bonn的，另一条是对应路由器Frankfurt的。星号“*”表示这台路由器没有IPv6路径，但存在链路。在示例10-50中显示了命令show isis ipv6 topology和show isis topology的输出，读者可以比较一下IPv6的拓扑和IPv4的拓扑。命令show isis topology显示了缺省的IPv4的拓扑。

示例10-50 在路由器Bonn上显示了IPv4的拓扑和IPv6的拓扑

img534_2



img535_1

如果在点到点链路上至少存在一个拓扑，就可以形成单个邻接关系。在示例10-51中使用命令**show clns is-neighbors detail**显示了路由器Bonn的接口Serial0/0.1上有一条与路由器Madrid之间的邻接，但是IPv4和IPv6拓扑都共享这个邻接。

示例10-51 在路由器**Bonn**和**Madrid**之间形成单个邻接，却由**IPv4**和**IPv6**拓扑共享这个邻接

img535_2

10.2.8 案例研究：层之间的路由泄漏

回忆当一台L1/L2路由器发送L1 LSP到一个区域时，它会通过在LSP中设置ATT位来通知其他的L1路由器它能够到达另一个区域。当需要转发数据包到区域外部时，L1路由器会把数据包转发到最近的L2路由器（到设置ATT位的路由器具有最小代价的路径）。

考虑图10-55中的网络。在区域1，路由器Prague和Bucharest都是L1/L2路由器。它们都是在L1更新数据包中设置了ATT位，并发送到路由器Belgrade和Zagreb，并且都是L1路由器。示例10-52中显示了路由器Belgrade的IS数据库。

示例10-52 区域1中的一台L1路由器的IS-IS数据库显示了两台都设置了ATT位的L1/L2路由器

img535_3

img536_1

img536_2

图10-55 在区域1中的网络具有多台L1/L2路由器，这将会导致从区域1到其他区域的路由选择没有效率

路由器Belgrade的IS-IS数据库显示了区域1中的所有路由器。路由器Prague和Bucharest都是L1/L2路由器，从而都设置了ATT位。回忆L1路由器使用最近的L1/L2路由器来转发业务量到区域外部。这两台L1/L2路由器到达路由器Belgrade具有相等的距离。路由器Belgrade的IPv4路由表（参见示例10-53）和IPv6路由表（参见示例10-54）显示了路由器Bucharest和Prague都被用来转发业务量到区域外部。

示例 10-53 L1路由器Belgrade上的IPv4路由表显示了到达区域外部的两条路径，表示具有两条缺省路由

img536_3

img537_1

示例**10-54** **L1**路由器**Belgrade**上的**IPv6**路由表显示了到达区域外部的两条路径，表示具有两条缺省路由

img537_2

正如读者在示例10-53和示例10-54中所看到的，路由器Belgrade的路由表中只有L1路由。所有区域外部的地址都跟在由缺省路由指定的路径后。缺省路由（IPv4: 0.0.0.0/0，IPv6: ::/0）显示了两个下一跳地址，一个来自路由器Bucharest（S0/0.3），一个来自路由器Prague（S0/0.1）。

地址172.16.23.0/24和2001:db8:0:17::/64是区域3中和路由器Warsaw相连的地址。对于路由器Belgrade来说，到达这些地址的最优路径是通过路由器Prague。但是，由于对于路由器Belgrade来说，路由器Bucharest和Prague是相等距离的L1/L2路由器，因此在路由器Belgrade发送业务量到地址172.16.23.0/24和2001:db8:0:17::/64时将会在这两台L1/L2路由器之间进行负载均分共享。一半的业务流量通过较优的路径，另一半则通过一条较长的路径。示例10-55中显示了路由器Belgrade上ping172.16.23.1和记录的路由，可以看出，一条路由通过Bucharest（172.16.11.2），而另一条路由通过Prague（172.16.22.2）。

示例10-55 使用带路由记录选项的ping显示了一个数据包从路由器Belgrade到Warsaw的往返路由

img537_3

img538_1

通过将172.16.23.0的地址通告到区域1的L1路由器，可以控制从路由器Belgrade到这个地址的业务流路径。这个把L2的路由通告给L1路由器的做法称为路由泄漏（route leaking）。下行方向的路由泄漏要求L1路由器需要明确地知道附加的地址和这些地址的路径。

路由泄漏的配置方式是，首先通过创建一个包含所要通告到该区域的地址列表，然后配置这个列表分发到L1路由器上。路由器Prague的配置参见示例10-56所示。

示例10-56 路由器Prague配置了路由泄漏，将L2的路由泄漏到L1区域

img538_2

通过访问列表编号100定义了所要通告的IPv4地址，另外通过名字为warsaw的前缀列表定义了需要通告的IPv6地址。**Redistribute isis ip** 命令检查匹配列表100的L2 IPv4地址，并将它们通告到该区域内的L1路由器上。在命令**address-family ipv6** 模式下使用**Redistribute isis** 命令检查匹配名字为warsaw的前缀列表的那些L2 IPv6地址，并将它们通告到L1区域内的L1路由器上。访问列表的讲述参见附录B，有关路由重新分配的更详细讲述参见第11章。

泄漏的路由通过和其他地址一样的方式进行通告：使用类型为128、130或135的IPv4可达性TLV，以及类型为236和237的IPv6可达性TLV。类型为128和130的TLV使用在窄幅度的度量，而类型为135的TLV使用在宽幅度的度量中。为了把泄漏的路由和其他相连接的IP路由或外部的IP路由区分开来，这里定义了一个Up/Down位，这个位的定义已经在前面的章节“跨域范围的前缀分发”中讲述过了。如果Up/Down位设置为0，那么这条路由就是始发于本地L1区域的。如果Up/Down位设置为1，那么

说明这条路由是由L2泄漏到L1的。设置该位有助于避免产生路由环路。一台L1/L2路由器将不会把设置了Up/Down位的L1路由地址通告给其他的L2路由器。

泄漏到L1的路由在路由表和IS-IS数据库中已有标识。在示例10-57和示例10-58中分别显示了路由器Belgrade的IPv4路由表和IPv6路由表。而在示例10-59中显示了路由器Belgrade的IS-IS数据库信息。


示例10-57 在路由器Belgrade的路由表中泄漏的路由被标识为“**ia**”路由，即区域间的路由

img539_1

示例**10-58** 在路由器**Belgrade**的路由表中泄漏的**IPv6**地址路由被标识为“**IA**”路由，即区域间的路由

img539_2

img540_1



示例**10-59** 在路由器**Belgrade**的IS-IS数据库中泄漏的地址被标识为“**IP-inter-area**”地址

img540_2

在路由表中，将IPv4的泄漏路由指定为“ia”类型，将IPv6的泄漏路由指定为“IA”类型，从而可以显示出这些IPv4和IPv6泄漏路由为IS-IS区域间路由。路由器把由路由泄漏发起的数据库条目显示为IP-Interarea和IPv6-

Interarea地址。

10.2.9 案例研究：多个L1区域运行在同一台路由器上

IOS软件系统具有在单台路由器上支持多个L1区域同时运行的能力：在IOS路由器上最多可配置29个L1区域，但是只能配置一个L2区域。在缺省情况下，第一个配置的IS-IS路由选择进程定义为L2区域。它也可以定义单个L1区域。为了在路由器上增加运行另外的L1区域，需要定义另外的IS-IS路由选择进程。在所配置的每一个L1区域内的路由器都是通过这台L1/L2路由器来通信的。因为每一个区域都不需要具有单一的L1/L2路由器到达网络的其他部分，因此L1/L2路由器的数量可以减到最少。

在图10-55的网络中，路由器Warsaw位于区域3。对于该区域，路由器Warsaw是一台L1/L2路由器。从路由器Warsaw到网络的其他部分的惟一连接是通过路由器Prague的。改变路由器Prague的配置，使Prague同时成为区域1和区域3的L2路由器。路由器Prague新的配置参见示例10-60。

示例10-60 路由器Prague配置了多个L1区域

img541_1

配置的第二个IS-IS路由选择进程命名为“three”。同时，把连接路由器Prague和Warsaw的接口Serial0/0.1指定运行IS-IS “three” 进程。IS-IS “three” 自动地配置为L1区域，这可以在路由器Prague上通过命令show clns neighbor看到，参见示例10-61所示。

示例10-61 在路由器**Prague**上配置了多个**L1**区域

img541_2

因为路由器Warsaw所有的邻居都出现在区域3内，因此它现在被配置作为L1路由器，参见示例10-62所示。在这个简单的网络中，这一步骤是不需要的，因为路由器Warsaw只有一个和路由器Prague相连的连接通向区域3外部，路由器Prague和路由器Warsaw只需要使用L1进行通信。在一个大型的网络中可能希望具有这一步骤，以便确保路由器Warsaw不会试图通过L2连接到其他路由器。

示例10-62 路由器Warsaw配置作为一台L1路由器

img541_3

示例10-63显示了路由器Warsaw作为一台L1/L2路由器时的IS-IS数据库和路由表。

示例10-64显示了路由器Warsaw作为一台L1路由器时的IS-IS数据库和路由表。

示例10-63 路由器Warsaw的数据库包含了区域3路由器和所有L2路由器的条目

img542_1

示例**10-64** 路由器**Warsaw**的数据库仅仅包含了区域**3**的路由器和链路，而它的路由表仅仅包含了不属于区域**3**的地址的缺省路由

img542_2

当路由器**Warsaw**作为区域**3**的L1/L2路由器时，它的IS数据库包含了区域**3**内每一台路由器的条目和每一台L2路由器的条目，因此那些L2路由器的所有地址都可以到达。它的路由表包含了网络中每一个可达的地址的路由条目。路由器**Warsaw**作为L1路由器时，明显地减小了它的IS-IS数据库和路由器的大小。

10.3 集成IS-IS协议的故障诊断

IS-IS协议故障诊断的基本方法和第8章讲述的OSPF协议的故障诊断方法十分相似。集成IS-IS协议和其他IP路由选择协议的故障诊断相比，一个主要的不同之处是IS-IS协议使用的是CLNS PDU数据包，而不是IP数据包。如果你是在对协议本身做故障诊断的话，记住你是在做CLNS协议的故障诊断，而不是IP协议。

正如所有的路由选择协议一样，故障诊断的第一步是检查路由表来获取精确的信息。如果一个预期的路由条目在路由表中丢失了或变得不正确，那么故障诊断剩下来的任务就是确定引起故障的源头了。

在检查过路由表之后，查看链路状态数据库就是获取故障诊断信息的一个最重要的来源。正如在第8章中所建议的，一个比较好的实际经验是，为每一个区域保存一份L1类型的链路状态数据库的拷贝，以及保存一份L2类型的链路状态数据库的拷贝。这些保存的数据库拷贝应该有规律地进行更新，并作为日常工作的一部分。这样在网络出现问题或错误时，这些保存的数据库拷贝将可以提供一个稳定状态的参考。在检查一台单独的路由器配置时，可以考虑以下问题：

- 在IS-IS协议配置下面的**net** 语句是否指定了正确的NET地址？在该路由器上配置的区域ID和系统ID是否正确无误？配置的NET地址是否符合所在网络上正在使用的CLNS编址约定？
- 是否在正确的接口上使用命令**ip router isis** 或**ipv6 router isis** 启动IS-IS协议？
- IP地址和子网掩码或前缀的配置是否正确？在一个集成IS-IS环境里检查这些配置显得加倍重要，因为配置错误的IP地址不会妨碍部分地建立一个IS-IS邻接关系。

10.3.1 IS-IS邻接关系的故障诊断

使用命令**show clns is-neighbors** 可以显示IS-IS的邻居表。缺省条件下显示的是整张邻居表，当然也可以指定显示一个具体接口的邻居表。从这个表中，我们可以看出所有预期的邻居是否都出现了？并且它们的类型

是否正确？为了获取更详细的信息，可以使用命令**show clns is-neighbors detail**来显示与每一个邻居相关联的区域地址和IP地址，以及每一个邻居的上线时间，等等。

在检查邻接关系时，可以考虑以下问题：

- 路由器的层（level）是否配置正确？L1路由器只能和L1与L1/L2类型的路由器建立邻接关系，而L2路由器只能和L2与L1/L2类型的路由器建立邻接关系。
- 是否这两台邻居路由器都在发送Hello数据包？Hello数据包的层（level）是否正确？Hello数据包包含的参数是否正确？调试命令**debug isis adj-pachets**是用来查看Hello数据包的一个比较有用的命令。
- 如果使用了认证，那么在邻居之间的口令和认证是否相同？记住区域（第1层）和域（第2层）认证不是验证普通的邻接关系的，它们只验证LSP的交换。
- 是否存在任何阻塞IS-IS或者CLNS协议的访问列表？

在图10-54中的路由器Bonn发生了一个变化。Bonn和Frankfurt不再能够从其他地点通过IPv4访问了。参考示例10-65，从路由器Bonn的IPv4路由表中显示，其中没有经过路由器Madrid（接口Serial0/0.1）学到的路由。

示例10-65 路由器Bonn的IPv4路由表显示了从路由器Frankfurt的接口FA0/0学到的IS-IS路由，但从路由器Madrid的接口Serial0/0.1没有学到路由

img544_1

命令**show clns is-neighbors** 的输出显示了一条从路由器Bonn到Madrid的现有邻接，如示例10-66所示。但是，请注意路由器Madrid的类型是IS而不是L1也不是L2。这表明该邻接出现了问题。使用调试命令**debug isis adj-packets** 可以给我们一点提示查找问题出在哪里，参见示例10-67所示。

示例10-66 路由器Bonn的CLNS IS-IS邻居表显示它到Madrid和Frankfurt的邻接，但是到Madrid的邻接出了一些问题

img544_2

示例10-67 利用调试命令 **debug isis adj-packets** 观察 **IS-IS Hello (IIHs)** 的详细信息。这个输出显示来自于图10-54的路由器 **Bonn**，串行接口 **Serial0/0.1** 的 **IP** 地址出现了问题

img544_3

回顾一下路由器 **Bonn** 上的 **IPv4** 地址，说明串行接口 **Serial0/0.1** 的 **IP** 地址被无意改变了。**IOS** 软件在建立邻接关系之前会检查 **TLV** 中的地址。如果这个接口地址 **TLV**（类型132）中的 **IPv4** 地址和接收它的接口不属于同一个子网，那么邻接关系会建立，但邻接关系的类型是 **IS** 而不是 **L1** 或 **L2**，这表明存在影响邻接关系的配置问题。改正了这个问题后，路由器 **Bonn** 的 **CLNS IS** 邻居表和 **IPv4** 路由表就正常了，参见示例10-68所示。

示例10-68 解决问题后的 **CLNS** 邻居表显示出邻接类型是 **L1**、**L2**，或

L1/L2。路由器**Bonn**的路由表显示了来自这两个邻接邻居的**IP**路由

img545_1

在问题解决后的路由器Bonn上执行命令`debug isis adj-packets`，输出信息参见示例10-69。该信息是在串行接口Serial0/0.1上接收的。

示例10-69 在配置正确的路由器Bonn上观察IS-IS Hello（IIHs）的详细信息

img545_2

如果没有在配置了IPv4地址的每一条链路上配置IPv6地址，类似的邻接问题也存在于单一拓扑模式下的IPv6中。当图10-54的网络配置成单一拓扑模式的IPv6，路由器Bonn与Frankfurt之间就出问题了。路由器Frankfurt不支持IPv6。示例10-48显示了路由器Bonn上 调试命令 **debug isis adj-packets** 的输出信息。调试信息说明存在一个链路本地地址的问题：“在来自接口FastEthernet0/0的LAN IIH内没有可用的IPv6链路本地地址”。路由器Bonn上的IOS软件在建立邻接关系之前会检查地址和地址簇的有效性。在单一拓扑模式下，如果在一台路由器上同时配置了IPv4与IPv6，那么每一条具有IPv4地址的链路都必须具有一个IPv6的地址，每一条具有IPv6地址的链路也必须具有一个IPv4的地址，同时邻居

路由器也必须具有有效的IPv4和IPv6地址。在Hello数据包中，类型232的TLV包含了IPv6的链路本地地址。在单一拓扑模式下，如果一台邻居路由器在它的Hello数据包中没有包含有效的IPv6链路本地地址，那么将会形成IS类型的邻接关系（再次表明配置有问题）。

log-adjacency-changes 是一个IS-IS进程配置模式的命令，该命令将会在日志中记录所有邻接关系的变化，日志可能在缓存区也可能发送到一台日志服务器上。仔细阅读这样的日志记录对处理过去某段时间发生的邻接问题是很有用的。

10.3.2 IS-IS链路状态数据库的故障诊断

IS-IS链路状态数据库的信息可以通过命令**show isis database** 来查看。如果一台路由器是L1/L2路由器，那么缺省情况下将会同时显示L1和L2类型的数据库。如果只需要查看其中一个数据库，可以使用**level-1** 或**level-2** 关键字。如果需要查看LSP更详尽的信息，可以使用**detail** 关键字。如果指定一个LSP ID，也可以查看单个LSP的信息，参见示例10-70所示。

示例10-70 这个LSP来自于图10-54中路由器Zurich的L2类型的数据库

img546_1

如果一个序列号显著地高于其他LSP的序列号，就可能表明这个区域不稳定或者是L2类型的骨干不稳定。另一个网络不稳定的提示是，某个LSP的抑制时间从来不会变得很小。如果怀疑网络不稳定，可以使用命

令**show isis spf-log** 列出这台路由器上最近执行的所有SPF计算。

在示例10-71中，显示了图10-54中路由器Geneva的SPF日志。在显示这个日志大约 3min后，除了每隔15min由数据库重新刷新触发的周期性的SPF计算，并没有显示其他内容。在那个时间段，频繁的SPF计算开始产生，这表明网络发生了频繁的变化。 [\[26\]](#)

示例10-71 这个**SPF**的日志记录揭示了图**10-54**中区域**1**的不稳定性

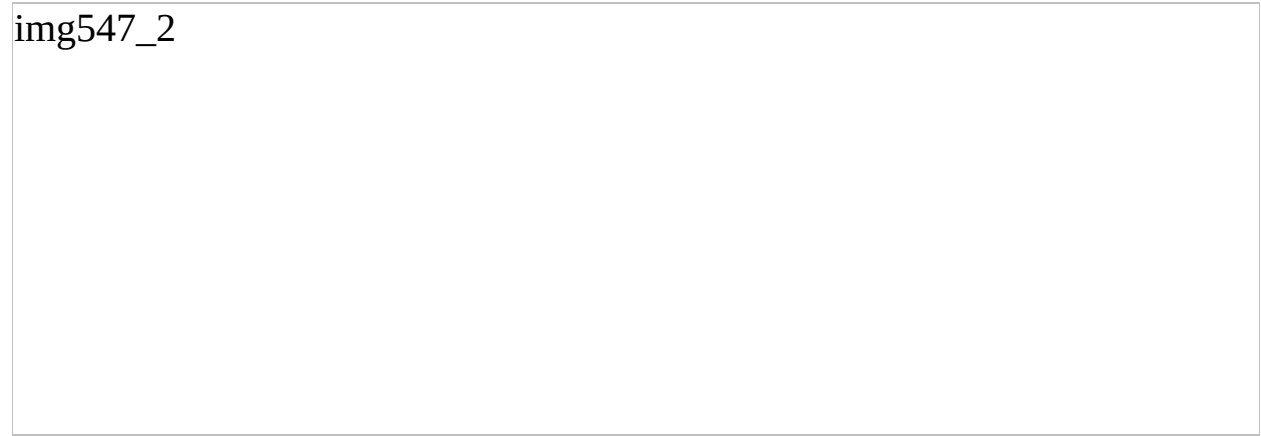
img547_1

为了进一步跟踪SPF日志暴露的网络震荡问题，还有3个有用的调试命令可以使用。示例10-72、示例10-73，以及示例10-74中显示了这3个调试命令的输出结果。在每一个输出中，从路由器Frankfurt的角度来看，调试信息都显示了图10-54中路由器Bonn的串行接口断开和重新连接的结

果。首先，**debug isis spf-triggers** 命令显示了与触发一个SPF计算有关的事件消息，参见示例10-72所示。第二条命令是**debug isis spf-events**，这个命令显示了触发事件引起SPF计算的一个详细报告，参见示例10-73所示。第三条命令是**debug isis spf-statistics**，它显示了有关SPF计算本身的信息，参见示例10-74所示。这里值得特别注意的是，这次进行的是完全的计算，这可能给路由器带来性能上的问题。

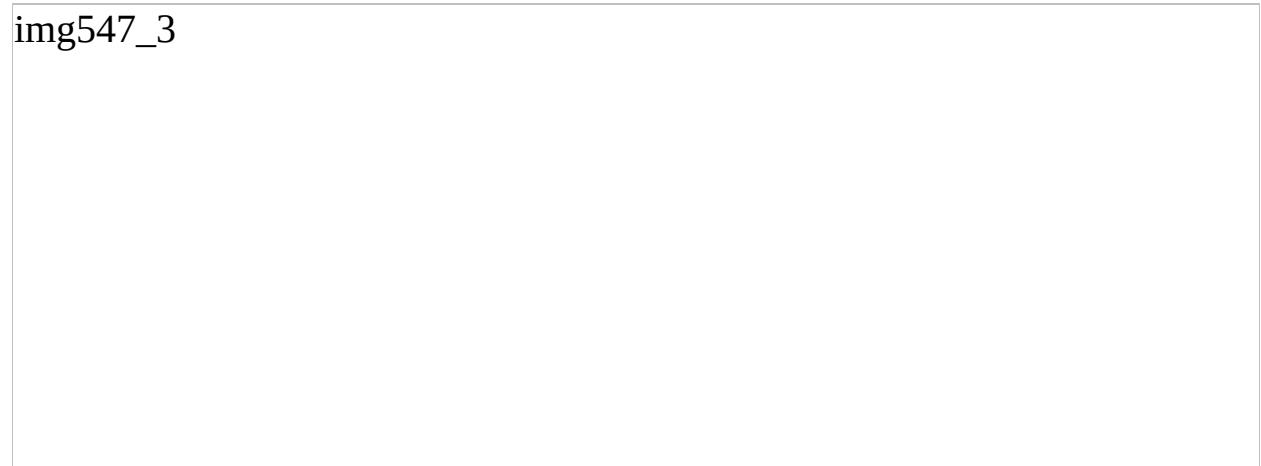
示例**10-72** **debug isis spf-triggers**命令显示了触发一个**SPF**计算的事件消息

img547_2



示例**10-73** **debug isis spf-events**显示了一个**SPF**计算的详细信息

img547_3



img548_1

示例10-74 **debug isis spf-statistics**显示了有关SPF计算本身的统计信息

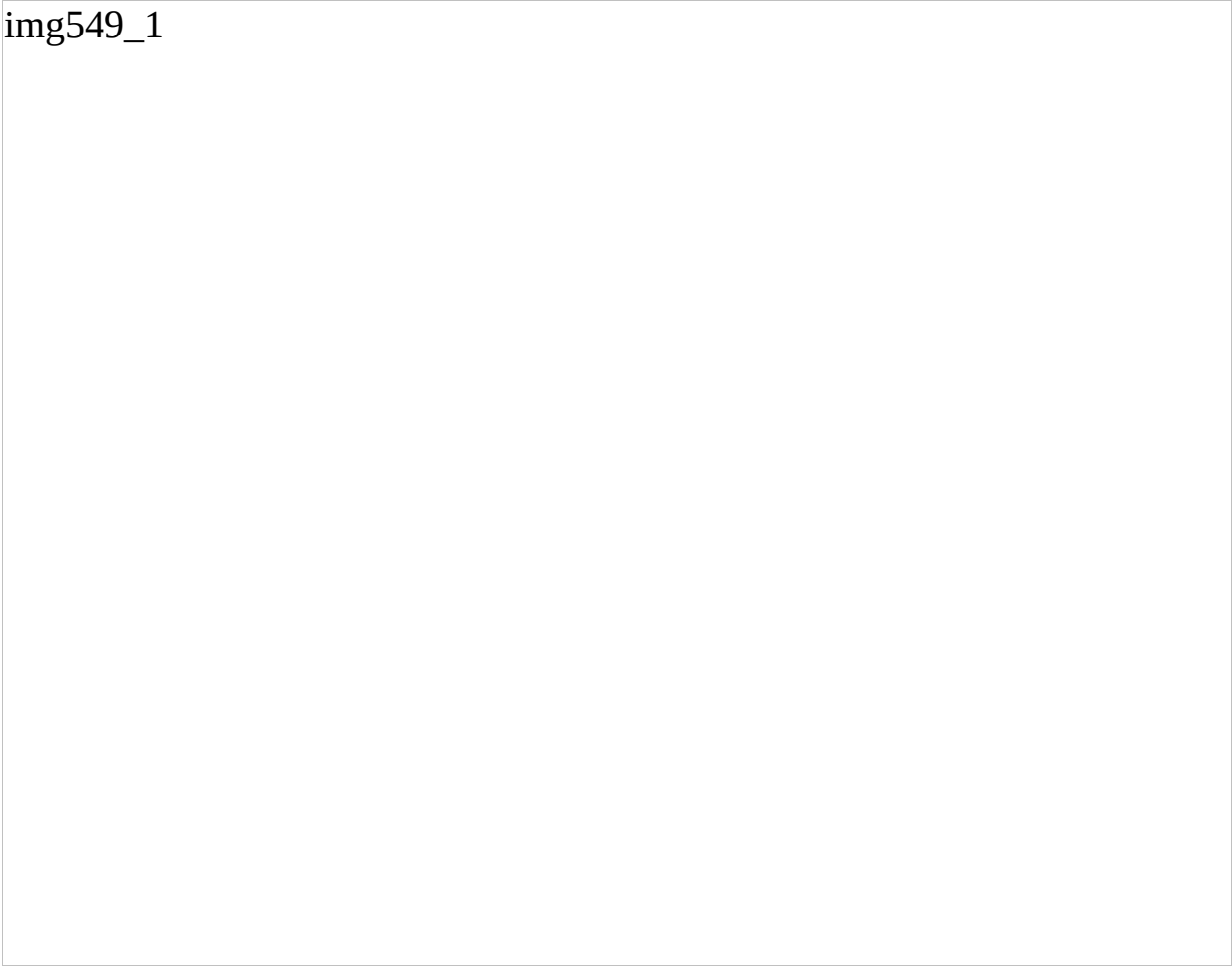
img548_2

在一个区域内的每一台路由器都必须维护一个同样的链路状态数据库。另外，IS-IS域内的每一台L1/L2和L2路由器都必须维护一个同样的L2类型的数据库。如果你怀疑某台路由器的链路状态数据库不能正确同步，可以检查它的LSP ID以其校验和。相同的LSP ID应该存在于每一个数据库中，并且在每一个数据库中的每一个LSP的校验和也都应该相同。

有两条调试命令可以帮助我们查看数据库的同步过程。第一条命令是**debug isis update-packets**，它显示了路由器接收和发送SNP与LSP的有关信息，参见示例10-75所示。第二条命令是**debug isis snp-packets**，它显示了路由器接收和发送某个指定的CSNP与PSNP的有关信息，参见示例10-76所示。

示例10-75 **debug isis update-packets**显示了路由器接收和发送SNP与LSP的有关信息

img549_1



示例**10-76** **debug isis snp-packets**显示了路由器接收和发送**CSNP**与**PSNP**的有关详细信息

img549_2

10.3.3 案例研究：运行于NBMA网络上的集成IS-IS

在图10-56中，显示了4台运行IS-IS协议的路由器，它们之间通过一个部分网状连接的帧中继网络相连。IP地址、DLCI和NET地址都标注在图上。所有路由器的IS-IS配置都已经确认是正确的，而且没有配置任何认证信息。

这个网络的故障是无法发现路由，参见示例10-77所示。邻居路由器的帧中继接口的IP地址可以ping通，但是邻居路由器上的其他接口地址都不能ping通，参见示例10-78所示。这些ping的结果表明帧中继PVC是正常运作的，IP也是工作的，但是路由器却没有进行路由选择。

img550_1

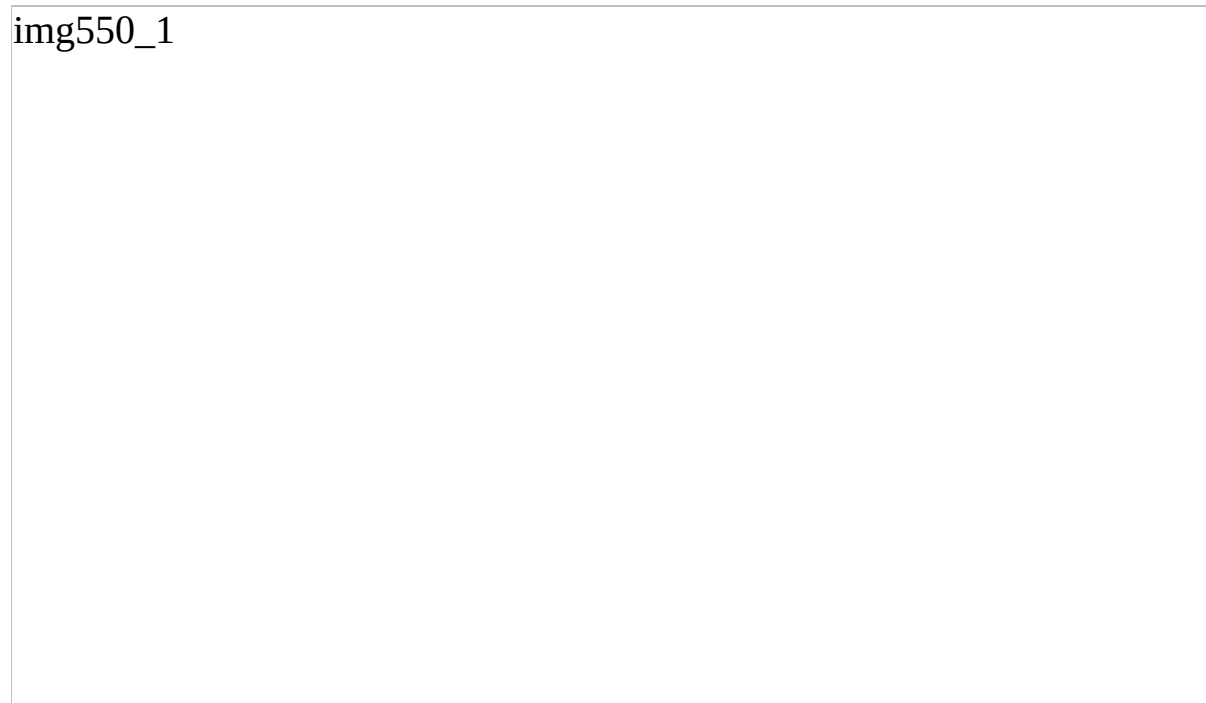

The diagram, which is missing from the page, would typically show four routers connected in a partial mesh topology over a Frame Relay network. It would likely include labels for IP addresses, DLCI values, and NET addresses for each router and its interfaces.

图10-56 IS-IS协议在帧中继网络上没有建立邻接关系

示例10-77 图10-56中路由器Oslo的路由表没有包含任何IS-IS路由

img550_2



示例**10-78** 与帧中继网络相连的其他接口可以**ping**通，但是路由器的可达地址却不能**ping**通

img550_3

img551_1

接下来的一步就是检查IS-IS的邻居表了。路由器Oslo的邻居表显示已经收到Hello数据包（参见示例10-79），而且这台路由器能够学习到邻居路由器的系统ID。同时，邻居表也显示了邻居路由器的IP地址和区域地址是正确的。但是，所有邻居路由器的状态都是Init，这表明还没有建立一个完全的邻接关系。查看一下链路状态数据库并确认不存在的邻接关系，路由器Oslo的数据库中的惟一LSP就是路由器本身的LSP，参见示例10-80所示。

示例10-79 路由器Oslo的IS-IS邻居表显示了可以收到Hello数据包，但是不能完全建立邻接关系

img551_2

示例**10-80** 路由器**Oslo**的链路状态数据库没有包含任何来自于邻居的**LSP**

img551_3

事实上，路由器正在接收Hello数据包，但邻接关系尚未建立成功，因此问题就指向了Hello数据包本身。如果Hello数据包中的参数不正确，这个PDU就会被丢弃。因此可以使用**debug isis adj-packets** 命令来观察Hello数据包。在这里需要特别注意的是，调试输出显示了“封装失败（encapsulation failed）”的消息，参见示例10-81所示。这些消息显示路由器显然不能解释收到的Hello数据包，因而丢弃了这些Hello数据包。

示例10-81 打开调试命令**debug isis adj-packets**，输出结果显示了由于封装失败而正在丢弃Hello数据包

img551_4

img552_1

任何时候看到封装失败的消息都应该怀疑数据链路的问题和所连接的接口问题。使用命令**show interface serial** 检查接口，没有发现存在严重的误码率，因此不像是帧中继PVC链路破坏了Hello数据包。下一步需要检查接口的配置。图10-56中的4台路由器的接口配置参见示例10-82～示例10-85。

示例**10-82** 路由器**Oslo**的接口配置

img552_2

示例**10-83** 路由器**Stockholm**的接口配置

img552_3

示例**10-84** 路由器**Copenhagen**的接口配置

img552_4

示例10-85 路由器Helsinki的接口配置

img553_1

通过比较这些配置发现了一个问题，虽然这个问题不一定是很明显。路由器Stockholm、Copenhagen和Helsinki都配置成了点到点的子接口，但路由器Oslo没有使用子接口。在缺省情况下，Cisco路由器的串行接口在封装成帧中继时是一个多点接口。因此，路由器Stockholm、Copenhagen和Helsinki发送点到点IS-IS Hello数据包，而路由器Oslo发送L1和L2类型的IS-IS LAN Hello数据包。

IS-IS协议的配置中没有一个类似于OSPF协议中的**ip ospf network** 命令的配置选项，因此，路由器Oslo必须重新配置成点到点子接口，并且必

须更改IP地址，以便使每一个PVC链路都在不同的子网中，参见示例10-86所示。

示例10-86 在路由器**Oslo**配置成点到点子接口后，并且进行重新编址以使每一个**PVC**成为一个单独的子网后，**IS-IS**路由选择就起作用了

img553_2

10.4 展 望

到现在为止，所有的IP IGP协议都已经论述完了，下一步将开始研究一些有效的工具来帮助控制我们的网络。第三部分“路由控制和互操作性”涵盖了路由再分配、缺省路由、按需路由选择、路由过滤和路由映射。

10.5 总结表：第10章命令总结

img553_3

img554_1

img555_1

10.6 复习题

1. 什么是中间系统？
2. 什么是网络协议数据单元？
3. L1、L2和L1/L2类型的路由器有什么不同？
4. 什么是Cisco路由器的缺省层？
5. 说明IS-IS区域和OSPF区域的不同之处。
6. 在具有相同区域ID的两台L1/L2路由器之间可以形成什么类型的邻接？如果L1/L2路由器的区域ID不同呢？
7. 在具有相同区域ID的两台L2-only路由器之间可以形成什么类型的邻接？如果L2-only路由器的区域ID不同呢？
8. 什么是网络实体标题（NET）？
9. 在NET中必须将NSAP选择符设置成什么值？
10. 系统ID的用途是什么？
11. 一台路由器是怎样确定它所在的区域的？
12. IS-IS协议在一个广播型子网上选取备份指定路由器吗？
13. 伪节点ID的用途是什么？
14. 一个IS-IS LSP缺省的最大老化时间（MaxAge）是多少？配置的MaxAge的最大值是多少？
15. OSPF协议老化它的LSA和IS-IS协议老化它的LSP的方式有什么基本的不同？
16. 一台IS-IS路由器多长时间重刷新一次它的LSP？

17. 什么是完全序列号数据包（CSNP）？它有什么用途？
18. 什么是部分序列号数据包（PSNP）？它有什么用途？
19. 过载位（OL）的用途是什么？
20. 区域关联位（ATT）的用途是什么？
21. Up/Down位的用途是什么？
22. ISO中规定的IS-IS度量有哪些？在Cisco IOS中支持哪些？
23. IS-IS的两种缺省度量类型是什么？它们的最大值是多少？
24. 一条IS-IS路由的最大度量值是多少？
25. L1类型的IS-IS度量和L2类型的IS-IS度量有什么不同？
26. 内部IS-IS度量和外部IS-IS度量有什么不同？
27. 在单一拓扑模式下，同时运行IPv4和IPv6的两台路由器之间的单条链路上可以创建几个邻接？如果是多拓扑模式呢？
28. 在单独一台路由器上可以有几个活动的L1区域？可以有几个活动L2区域？
29. 除了Inactive外，另外两个mesh组模式是什么？它们各有何利弊？

10.7 配置练习

1. 表10-8中显示了11台路由器的接口、接口地址和子网掩码。这个表还指定了属于同一个区域的路由器。使用下面的指导策略，写出每台路由器的集成IS-IS的配置。

- 为每台路由器配置自己的系统ID;
- 使用尽可能短的NET地址;
- 适当地把这些路由器配置成L1、L2或者L1/L2类型的路由器。

提示：首先画一张路由器和子网的图形。

表10-8 配置练习1~5的路由器信息

img556_1

img557_1

2. 在每一台路由器上配置多拓扑模式的IPv6。使用地址2001:db8:0:x::/64，这里的x是每一条链路上对应于IPv4地址的子网（只有第4个八位组）的十六进制值。
3. 在表10-8的区域2内的所有路由器上配置认证。路由器D和路由器E之间使用口令“Eiffel”，路由器E和路由器F之间使用口令“Tower”。使用的认证是md5类型。

4. 在表10-8中区域1上配置L1类型的认证，使用口令“Scotland”。使用没有钥匙链的明文口令。
5. 在表10-8的路由器上配置L2类型的认证，使用口令“Vienna”。使用带有钥匙链的明文口令。
6. 配置表10-8中区域0、区域1和区域3的L1/L2路由器，同时汇总它们IPv4和IPv6的L1地址。

10.8 故障诊断练习

1. 在示例10-87和示例10-88中，显示了路由器A和路由器B的IS-IS邻居表，这两台路由器是通过一个串行接口相连的。出现了什么错误？
2. 在示例10-89中，显示了一台路由器的调试信息，这台路由器没有和它TOO接口上的邻居路由器成功建立邻接关系。出现了什么错误？

示例**10-87** 故障诊断练习**1**，路由器**A**的**IS-IS**邻居表

img557_2

img558_1

示例**10-88** 故障诊断练习**1**，路由器**B**的**IS-IS**邻居表

img558_2

示例**10-89** 故障诊断练习**2**的调试输出信息

img558_3



[1] 国际标准化组织, “中间系统到中间系统域内路由选择信息交换协议, 提供无连接模型网络服务 (ISO 8473)”ISO/IEC 10589, 1992年。

[2] Ross Callon, “Use of OSI IS-IS for Routing in TCP/IP and Dual Environments, ”RFC 1195, 1990年12月。

[3] 无连接模型的网络服务——CLNP的网络层协议。

[4] Christian Huitema, Routing in the Internet, Prentice Hall PTR, Englewood Cliffs, NJ, 1995年。

[5] Radia Perlman, Interconnections: Bridges and Routers, Addison-Wesley, Reading, MA, 1992年。

[6] 对于某些共同术语的ISO / 欧洲拼法, 如“routeing”与“neighbour”是可以不用的。

[7] 在某些文档中, 例如RFC 1195, 把LSP定义为一个链路状态数据包 (Link State Packet) 。

[8] 这里实际上是4个ATT位, 并与不同的度量相关联。这些位的进一步解释在10.1.4小节中介绍。

[9] L2区域用于为一个单独的区域提供必要的到区域外部的出口, 新的区域可以作为L1类型, 并简单地连接到已经存在的L2区域上。

[10] 国际标准化组织, “Network Service Definition Addendum 2: Network Layer Addressing”, ISO/IEC 8348/Add.2, 1988年。

[11] Cisco公司的IS-IS实现需要一个6个八位组字节的系统ID。

[12] GOSIP Advanced Requirements Group (GOSIP高级需求组), “Government Open Systems Interconnection Profile (GOSIP) Version 2.0 [Final Text]”, Federal Information Processing Standard (联邦信息处理标准), U.S. Department of Commerce (美国商务部), National Institute of Standards and Technology (美国国家标准和技术协会), 1990年10月。

[13] Richard Colella, Ella Gardner和Ross Callon, “Guidelines for OSI NSAP Allocation in the Internet”, RFC 1237, 1991年7月。

[14] 国际标准化组织, “Internal Organisation of the Network Layer”, ISO 8648, 1990年。

[15] 有一个例外是, 路由器收到了一条比它链路状态数据库中相同LSP的实例更旧的LSP实例。在这种情况下, 该路由器将回复一个新的LSP。

[16] 国际标准化组织, “Protocol Identification in the Network Layer”, ISO/IEC TR 9577, 1990年。

[17] 读者已经在前面第7章中讲述EIGRP TLV的概念时熟悉了TLV的概念。事实上, 在RFC 1195中, 也把集成IS-IS TLV称为TLV。

[18] 在ISO中使用术语“代码”, 而在IETF中使用术语“类型”。由于“TLV”已经变得比较通用, 因此本章通常使用术语“类型”。这与本书第一版不同, 在第一版中作者选用了术语“代码”。

[19] 作为提示, RFC 1195规定了认证信息TLV的类型代码是133。而Cisco路由器使用ISO规定的类型代码10来标识认证信息TLV。

[20] 初级中学的数学教师喜欢用“语言问题”难为他们的学生, 因此在这里等价的表达方式是: 除了始发LSP的路由器, 每一台路由器都会扩散_n-2条不必要的LSP。

[21] **router isis** 命令也可以给一个名字, 例如router isis Warsaw。如果IS-IS和ISO-IGRP协议配置在同一台路由器上, 那么必须为其中一个进程或者同时为两个进程命名。如果没有配置ISO-IGRP协议, 则命名不是必须的。

[22] 正如前面讲到的, 这个LSP ID后面的星号表示该LSP是由这台路由器本身始发的。

[23] 请注意, 路由器Madrid由于没有L1类型的邻居, 因而配置成一台L2路由器。

[24] 路由器之间的口令或钥匙串都必须是相同的。它们是区分大小写的。也就是说，空格和其他字符也是口令的一部分。如果配置文件是免费在线创建而粘贴到路由器中的，就要小心尾部的空格。

[25] 在IOS软件的12.2（15）T、12.2（18）S、12.0（26）S、12.3、12.3（2）T，以及以后的版本支持多拓扑模式。

[26] 开始的4个触发事件是由路由器Zurich的串行接口几次状态变化引起的，接下来的3个事件是由于删除和接着错误配置了链路口令引起的，最后一个事件是在配置了正确的口令时引起的。

第三部分

路由控制和互操作性

第11章 路由重新分配

第12章 缺省路由和按需路由选择

第13章 路由过滤

第14章 路由映射

本章包括以下主题：

- 重新分配的原则；
- 配置重新分配。

第11章

路由重新分配

当路由器使用路由选择协议通告从其他方式学习到的路由时，路由器将执行重新分配。这里所谓的其他方式可能是另外一个路由选择协议、静态路由或直连目标网络。例如，路由器可能同时运行OSPF进程和RIP进程。如果设置OSPF进程通告来自RIP进程的路由，这就叫做重新分配RIP。

在整个IP网络中，如果从配置管理和故障管理的角度看，我们通常更愿意运行一种路由选择协议，而不是多种路由选择协议。然而，现代的网络又常常迫使我们接受多协议IP路由选择域这一现实。当部门、分公司乃至整个公司合并时，必须统一它们原来的自主网络。

在大部分案例中，将要被合并的网络在实现和发展上都不相同，它们满足不同的需求，是不同设计理念的产物。这种差异性使得向单一路由选择协议的迁移成为一项复杂的任务。在某些案例中，公司的策略可能会强制使用多种路由选择协议。而在少数场合还会出现因网络管理员不能很好地协同工作而采用多种路由选择协议。

多厂商环境是需要重新分配路由的另一个因素。例如，一个运行Cisco EIGRP的网络可能会与使用另一个厂商路由器的网络合并，而这种路由器仅支持RIP和OSPF。如果不进行重新分配，那么Cisco路由器需要使用一种公开的协议重新配置或者用Cisco路由器替代非Cisco路由器。

当多种路由选择协议“被拼凑”在一起时，使用重新分配是很有必要的，而且重新分配也是一个严谨网络设计的一部分。图11-1给出了一个例子，这里把两个OSPF进程域连接在一起，但是OSPF进程之间并不直接通信，取而代之的是在每台路由器上配置静态路由，指向其他OSPF域内的被选网络。

例如，路由器Spalding包含指向网络192.168.11.0和192.168.12.0的路由，Spalding把这些静态路由重新分配到OSPF中，OSPF又向OSPF 10内的其他路由器通告这些路由。这样做的结果是向OSPF 10隐瞒了OSPF 20中

的其他网络。重新分配使得OSPF的动态特性和静态路由的精确控制性融合在一起。

img564_1

图11-1 在这个网络中，在每台路由器上都配置了静态路由，并且向OSPF重新分配这些路由。结果是，可以做到精确地控制在两个OSPF域之间的前缀通告

如果不是非要使用动态路由选择协议的话，在拨号环境中向动态路由选择协议重新分配静态路由也是非常有用的。动态协议周期性的管理流量会导致拨号线路始终保持接通状态。通过阻止路由更新和Hello信息通过线路，并在两边配置静态路由，管理员可以确保线路只在有用户流量通过时才接通。向动态路由选择协议重新分配静态路由，可以使拨号线路两边的所有路由器都知道链路对方的所有网络。

注意到，除了少数特例外，[\[1\]](#)在相同路由器上存在不止一种路由选择协议并不意味着重新分配自动发生。重新分配必须被明确地配置。在没有使用重新分配的单一路由器上配置多种路由选择协议的方法叫做午夜航船（Ships In the Night, SIN）路由选择。路由器将会在每个进程域内向它的对等路由器传递路由，但是进程域之间却一无所知——这就好比黑暗中航行的船只一样。虽然SIN路由选择法通常指在相同路由器上多种路由选择协议为多种可路由协议进行路由选择（例如OSPF为IP和NLSP为IPX进行路由选择），但是它也可以指在单一路由器上两个IP协议为单独的IP域进行路由选择。

11.1 重新分配的原则

IP路由选择协议的能力相差非常大。对重新分配影响最大的协议特性是度量和距离的差异性，以及每种协议的有类别和无类别能力。在重新分配时如果忽略了对这些差异性的考虑将导致以下后果，最好情况会出现某些或全部路由交换失败，最坏情况将造成路由环路和黑洞。

11.1.1 度量

图11-1中的路由器正在向OSPF重新分配静态路由，然后它们会向其他OSPF路由器通告这些路由。虽然静态路由没有相关联的度量，但每条OSPF路由必须有一个代价值。有关度量冲突的另一个例子是向EIGRP重新分配RIP路由，RIP的度量是跳数，而EIGRP使用带宽和时延。在这两种情况中，接收被重新分配路由的协议必须能够将自己的度量与这些路由关联起来。

所以执行重新分配的路由器必须为被重新分配的路由指派度量。图11-2给出了一个例子，这里EIGRP被重新分配到OSPF，同时OSPF也被重新分配到EIGRP。OSPF不能理解EIGRP的复合度量，EIGRP也不能理解OSPF的代价。因此在重新分配进程中，路由器必须在向OSPF传递EIGRP路由之前为每一条EIGRP路由分配代价度量。同样的，路由器在向EIGRP传递OSPF路由之前也必须为每一条OSPF路由分配带宽、时延、可靠性、负载和MTU度量值。如果分配了不正确的度量，重新分配将会失败。

img565_1




图11-2 当重新分配路由时，必须为路由分配一个接收协议可以理解的度量值

本章后面的案例分析将会讨论为了达到路由重新分配的目标，应该如何配置路由器用来分配度量。

11.1.2 管理距离

度量的差异性产生了另一个问题：如果路由器正在运行多个路由选择协议，并从每个协议都学习到一条到达相同目标网络的路由，那么应该选

择哪一条路由呢？每一个路由选择协议均使用自己的度量方案定义最优路径。比较不同度量的路由，例如代价和跳数，就像比较苹果和橙子一样。

这个问题的答案是管理距离。正像为路由分配度量以便可以确定首选路由一样，我们可以向路由源分配管理距离值以便确定首选路由源。管理距离被看作是一个可信度测度，管理距离越小，协议的可信度越高。

例如，假设运行RIP和EIGRP的路由器从邻居RIP路由器那里学习到一条指向网络192.168.5.0的路由，从邻居EIGRP路由器那里学习到一条指向相同网络的路由。由于EIGRP的复合度量，使得该协议更有可能确定最佳路由。因此，EIGRP比RIP更可信。

表11-1列出了缺省的Cisco管理距离。EIGRP的管理距离为90，而RIP的管理距离为120。因此，可以认为EIGRP比RIP更值得信赖。

表11-1 **Cisco缺省管理距离**

路由源	管理距离
直连接口*	0
静态路由	1
EIGRP汇总路由	5
外部BGP	20
EIGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EGP	140
外部EIGRP	170
内部BGP	200
未知	255

*回忆一下第3章，当静态路由使用接口替代下一跳地址时，目的网络即被认为是直连网络。

虽然管理距离帮助解决了不同度量带来的混乱，但是它又为重新分配带来了问题。例如，在图11-3中，Gehrig和Ruth都在向IGRP重新分配RIP路由。Gehrig通过RIP学习到网络192.168.1.0，并且将其通告给IGRP域。结果是，Ruth不仅通过RIP从Combs处学习到网络192.168.1.0，而且还通过IGRP从Meusel那里也学习到该网络。

示例11-1给出了Ruth的路由表。注意，指向网络192.168.1.0的路由是一条IGRP路由。Ruth之所以选择IGRP路由是因为IGRP比RIP具有更小的管理距离。Ruth将经过Meusel沿着这条“风景优美的路线”发送所有数据包，代替直接向Combs发送数据包。

水平分隔阻止了在图11-3的网络中路由环路的发生。Gehrig和Ruth最初都向IGRP域通告网络192.168.1.0，并且最终4台IGRP路由器都收敛到一条到达该网络的路径。然而，这种收敛是不可预知的。重新启动Lazzeri和Meusel可以看到这一情形。在重新启动后，Ruth的路由表显示到达网络192.168.1.0的下一跳路由器是Combs（参见示例11-2）。

img566_1

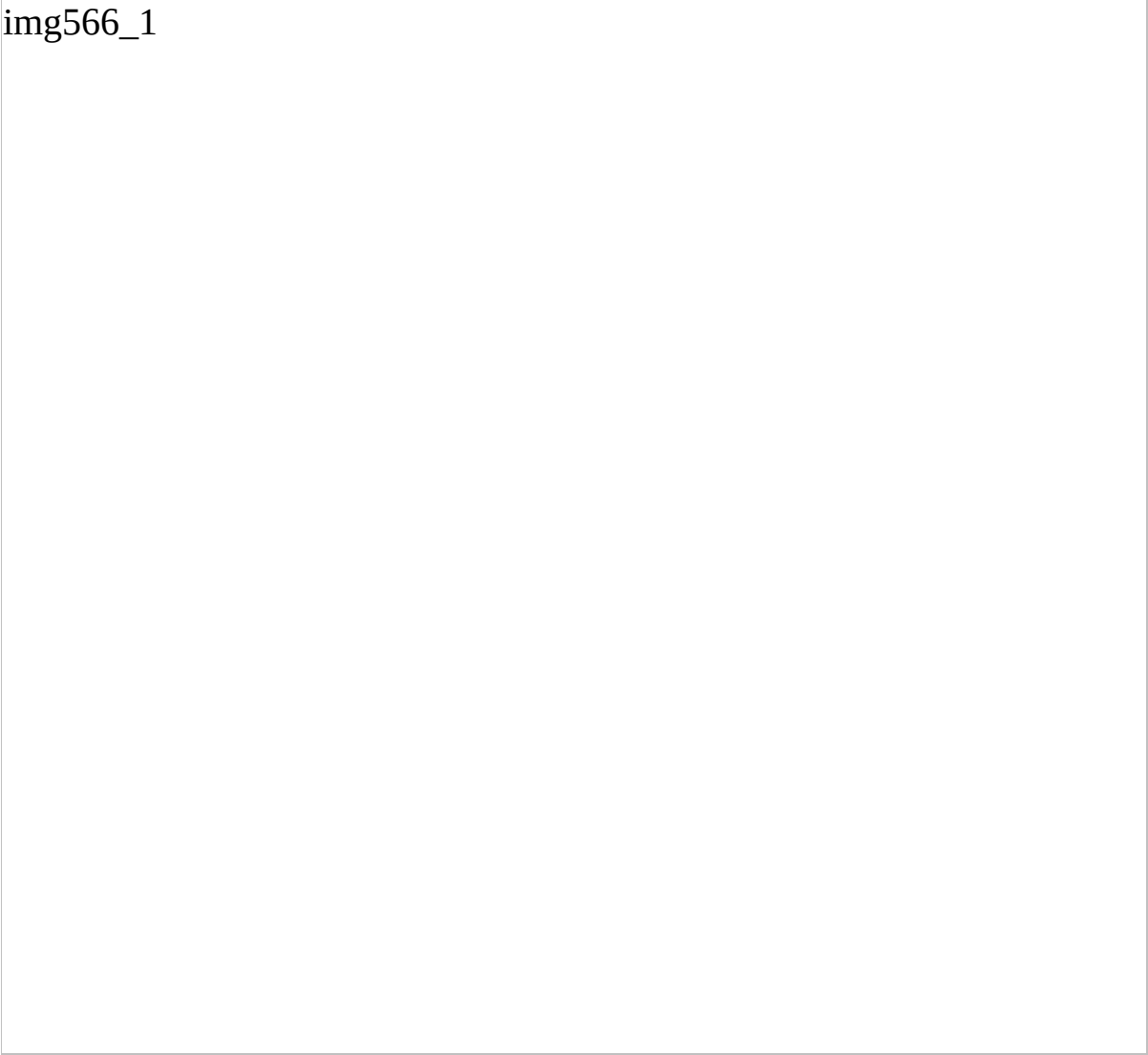


图11-3 通过RIP和IGRP向Ruth通告网络192.168.1.0

示例**11-1** 虽然从**Ruth**到达网络**192.168.1.0**的最佳路径是从**SO**出发经过**Combs**去往目标网络，但是**Ruth**却选择了从**S1**出发经过**Meusel**的路径

img567_1

示例**11-2** 图**11-3**中的网络收敛不可预知。在路由器重新启动后，**Ruth**现在选择经过**Combs**到达网络**192.168.1.0**

img567_2

在重新启动后收敛不仅难以预知，而且很慢。示例11-3显示了重新启动完毕大约又经过3min后Gehrig的路由表。它使用Lazzeri作为到达网络192.168.1.0的下一跳路由器，但是ping该网络中的一个在线地址却发生失败。从Lazzeri的路由表（参见示例11-4）可以看出问题所在：Lazzeri使用Gehrig作为下一跳路由器，因此存在路由环路。

示例11-3 重新启动后不久，**Gehrig**为数据包选择经**Lazzeri**到达网络**192.168.1.0**的路径

img567_3

示例**11-4** **Lazzeri**路由为数据包选择经**Gerig**到达**192.168.1.0**的路径，因而产生路由环路。注意路由的年龄

img568_1

下面是导致环路的事件顺序：

1. 当Lazzeri和Meusel重新启动时，Gehrig和Ruth的路由条目显示经过Combs可以到达网络192.168.1.0。
2. 随着Lazzeri和Meusel启动完毕，Gehrig和Ruth发送包括网络192.168.1.0的IGRP更新信息，仅仅由于运气，Ruth比Gehrig更早一点发送了更新信息。
3. Meuse接收到Ruth的更新信息，把Ruth作为下一跳路由器，并且向Lazzeri发送更新信息。
4. Lazzeri接收到Meusel的更新信息，把Meusel作为下一跳路由器。
5. Lazzeri和Gehrig在差不多相同的时刻互相发送了更新信息。Lazzeri把Gehrig作为到达网络192.168.1.0的下一跳路由器，因为这条路由比

Meusel的路由更接近目标。Gehrig把Lazzeri作为到达网络192.168.1.0的下一跳路由器，因为Lazzeri的IGRP通告的管理距离比Combs的RIP通告的小。至此环路产生。

水平分隔和失效计时器最终将会解决这一问题。虽然Lazzeri正在向Meusel通告192.168.1.0，但是Meusel仍会继续使用经过Ruth的更近的路径。由于Ruth是下一跳路由器，所以在Meusel的接口S1上，水平分隔对192.168.1.0有效。Meusel还向Lazzeri通告192.168.1.0，但是Lazzeri认为Gehrig更接近目的网络。

由于Lazzeri和Gehrig都将对方看作是去往192.168.1.0的下一跳路由器，所以它们相互不通告此路由。保存在它们路由表中的这条路由一直老化到失效计时器超时为止（参见示例11-5）。

示例11-5 在指向**192.168.1.0**路由的失效计时器超时后，这条路由将会被声明为不可达，并且抑制计时器被启动


img568_2

img569_1

在Lazzeri的失效计时器超时后，指向192.168.1.0的路由将被抑制。虽然Meusel正在通告指向该网络的路由，但是Lazzeri直到抑制计时器超时才能接收该通告。示例11-6显示出Lazzeri最终接收了这条来自Meusel的路由，示例11-7显示出Gehrig通过Lazzeri可以成功地到达192.168.1.0。但是这两台路由器花费了9min时间才得以收敛，而且还使用了一条非最佳路由。

示例 11-6 在网络**192.168.1.0**的抑制计时器超时后，**Lazzeri**接受**Meusel**通告的路由

img569_2



示例**11-7** **Gehrig**现在可以通过**Lazzeri**到达网络**192.168.1.0**

img569_3

管理距离导致的问题会比前面例子所述的非最佳路径、不可预知的行为以及慢收敛问题更严重。例如，图11-4给出的网络与图11-3中的网络本质上是相同，除了IGRP路由器之间的链路为帧中继永久虚电路。在缺省情况下帧中继接口上的IP水平分隔功能是关闭的，结果是在Lazzeri和Gehrig之间以及Meuse和Ruth之间会形成永久路由选择环路，并且从IGRP域无法到达网络192.168.1.0。

img570_1

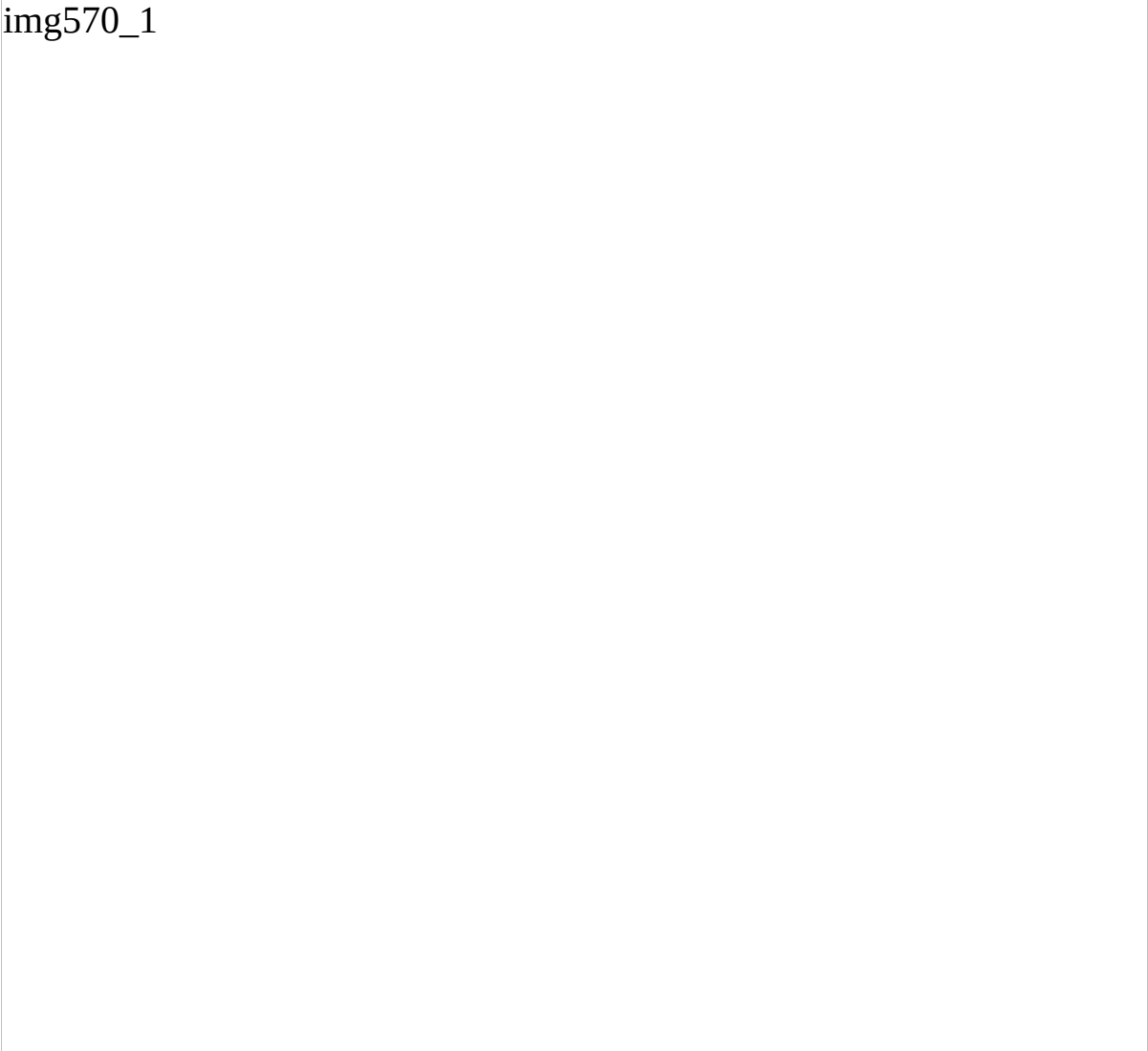


图11-4 因为在帧中继接口上水平分隔功能在缺省状态下是关闭的，所以在这个网络上将会形成永久路由选择环路

在重新分配时有几种工具和策略可以避免路由选择环路，比如使用操作管理距离、路由过滤和路由映射。第13章讨论路由过滤，第14章讨论路由映射。这两章还示范了修改管理距离的技术。

11.1.3 从无类别协议向有类别协议重新分配

从无类别路由选择进程域向有类别域重新分配路由会产生哪些影响，这值得我们仔细地考虑。为了理解为什么这样做，首先有必要理解有类别路由选择协议怎样应对变长子网划分。回想第5章的内容，有类别路由

选择协议不能通告携带子网掩码的路由。对于有类别路由器所接收到的每一条路由，无外乎是下面两种情况之一：

- 路由器将有一个或多个接口连接到主网上；
- 路由器将没有接口连接到主网上。

在第一种情况下，为了正确地确定数据包目标地址的子网，路由器必须使用它自己的主网掩码。在第二种情况下，公告信息中仅包含主网地址，因为路由器不知道使用哪一个子网掩码。

图11-5给出了路由器有4个接口分别连接到192.168.100.0的各子网上。该网络采用了变长子网划分——两个接口的子网掩码为27位，另两个为30位。如果路由器运行有类别协议，例如IGRP，那么它将不能从27位掩码推出30位掩码的子网，并且也不能从30位掩码推出27位掩码的子网。那么，协议如何处理冲突的掩码呢？

img570_2

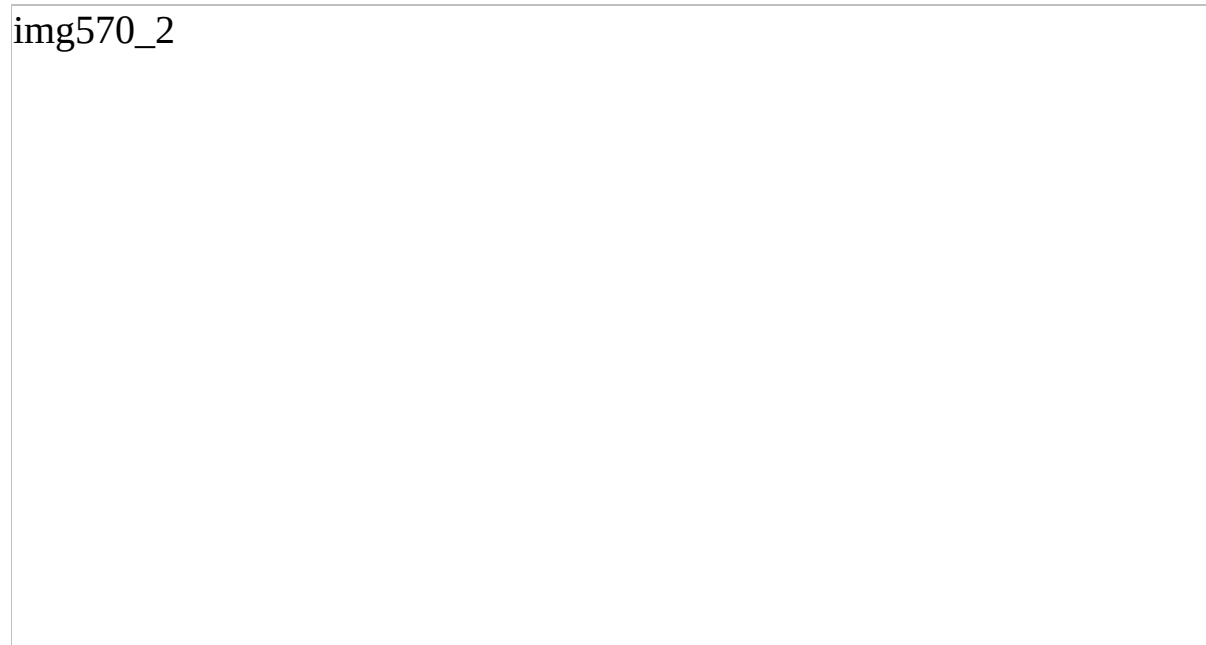



图11-5 如果路由器运行有类别路由选择协议，它应该选择什么掩码呢

在示例11-8中，使用调试手段观察图11-5中路由器发出的IGRP通告。注意，子网192.168.100.128/27的通告从接口E0发出，网络使用了27位掩码，但是没有从该接口发送192.168.100.4/30和192.168.100.8/30通告。类似的，192.168.100.8/30的通告从接口S1发出，掩码为30位，但是没有从

该接口发送192.168.100.96/27和192.168.100.128/27通告。相同的情形同样适用于所有4个接口。在192.168.100.0的子网中，仅那些掩码与接口掩码相同的子网才会从此接口通告。最后结果是接口E0和E1的IGRP邻居路由器不知道掩码为30位的子网，接口S0和S1的IGRP邻居路由器也不知道掩码为27位的子网。

示例11-8 有类别路由选择协议将不在掩码一致的接口之间通告路由

img571_1



当从无类别路由选择协议向有类别路由选择协议重新分配路由时，仅在掩码相同的接口之间通告路由这一特性将得到应用。在图11-6中，OSPF域的子网是经过变长子网划分得到的，Paige将来自OSPF的路由信息向IGRP重新分配。

img571_2




图11-6 Paige将来自OSPF的路由重新分配进IGRP

如示例11-9所示，Paige知道OSPF和IGRP域内的所有子网。因为OSPF是无类别协议，所以路由器知道连接到Gibson的每个子网的相应掩码。由于Paige的IGRP进程使用24位掩码，因此172.20.113.192/26和172.20.114.48/28不一致，所以不能被通告（参见示例11-10）。注意，IGRP对172.20.112.0/24和172.20.115.0/24进行通告，结果在OSPF域内Leonard仅知道掩码为24位的子网（参见示例11-11）。

示例11-9 Paige知道图11-6中的所有6个子网，它们不是来自OSPF、IGRP，就是直接连接

img571_3

img572_1

示例**11-10** 在来自**OSPF**的路由信息中，仅掩码为**24**位的路由被成功地重新分配进入**IGRP**域，该**IGRP**域也使用**24**位掩码

img572_2

示例**11-11** **Leonard**仅知道掩码为**24**位且在**OSPF**域内的子网，从**Leonard**这里不能到达网络**172.20.113.192/26**和**172.20.114.48/28**

img572_3

配置部分包括的案例研究将示范从无类别路由选择协议向有类别路由选择协议可靠地进行重新分配的方法。

11.2 配置重新分配

配置重新分配分为两步：

步骤1： 在路由选择协议中配置接收重新分配的路由，其中使用命令redistribute指定路由源点。

步骤2： 为重新分配的路由指定度量值。

图11-7中Lajoie的EIGRP配置见示例11-12。

示例11-12 Lajoie的EIGRP配置显示出OSPF的路由被重新分配到EIGRP进程

img572_4

img573_1

图11-7 为简单网络配置路由重新分配

示例11-12中的配置把OSPF进程1发现的路由向EIGRP进程1重新分配。命令的Metric部分为路由分配了EIGRP度量值。按照顺序，命令中各数字分别表示：

- 带宽，单位是kbit/s；
- 时延，单位是10μs；
- 可靠性，为255的若干分之一；
- 负载，为255的若干分之一；
- MTU，单位为八位组字节。

Lajoie的OSPF配置见示例11-13。

示例11-13 Lajoie的OSPF配置显示出EIGRP的路由被重新分配到OSPF

img573_2

上面的配置将EIGRP进程1发现的路由重新分配到OSPF进程1。命令的Metric部分为每一条被重新分配的路由分配了OSPF代价值30。重新分配使得Lajoie成为OSPF域的ASBR，并且被重新分配的路由是作为外部路由进行通告的。命令的**metric-type** 部分指明了外部路由的类型为E1。关键字**subnets** 仅当向OSPF重新分配路由时使用，它指明子网的细节将被重新分配；没有它，仅重新分配主网地址。在案例研究中会更多地讨论关键字**subnets** 。

另一种分配度量的方法是使用**default-metric** 命令。例如，前面的OSPF配置也可以改写为示例11-14的形式：

示例11-14 在Lajoie的OSPF配置中使用**default-metric**命令为所有被重新分配到OSPF的路由指定OSPF度量

img573_3

该配置同前面配置所产生的结果完全相同。当重新分配来自多个源点的路由时，命令**default-metric**显得十分有用。例如在图11-7中，假设路由器Lajoie不仅运行EIGRP和OSPF，而且还运行RIP。相应的OSPF配置见示例11-15。

示例11-15 对于所有被重新分配到OSPF的EIGRP和RIP路由，在Lajoie的OSPF配置中使用**default-metric**命令为它们指定OSPF度量

img573_4

在上面的配置中，对于所有来自EIGRP和RIP的路由，所分配的度量均为OSPF代价30。

在这里，两种分配度量的方法也可以相互使用。例如，假设置Lajoie向EIGRP重新分配OSPF和RIP路由，但要求对RIP路由进行通告时，所使用的度量要不同于OSPF，相应的配置见示例11-16。

示例11-16 Lajoie的EIGRP配置使用带度量参数的**redistribution**命令为从RIP重新分配来的地址指定度量值，而使用**default metric**命令对所有其他重新分配的地址分配EIGRP度量

img574_1

在上面的配置中，使用命令**redistribute** 中关键字**metric** 分配度量值优于**default-metric** 命令分配度量值。来自RIP的路由被通告到EIGRP，这些路由所使用的度量在配置行**redistribute rip** 中指明。而来自OSPF的路由则使用**default-metric** 命令指定的度量被通告。

如果关键字**metric** 和命令**default-metric** 都没有指定度量，那么被重新分配到OSPF的路由的度量缺省值为20，而其他协议路由度量的缺省值为0。IS-IS可以理解0度量，但是RIP不能，因为它的跳数在1到16之间。0度量与EIGRP的多度量格式也不兼容。因此这两种协议都必须为重新分配的路由分配合适的度量，否则重新分配将不能进行。下面的案例研究将会分析向各种IP IGP重新分配路由的配置方法。此外，案例研究中还更多地安排了关于从有类别向有类别、从无类别向无类别以及从无类别向有类别重新分配路由等常见问题的分析。

11.2.1 案例研究：重新分配IGRP和RIP

在图11-8的网络中，Ford运行IGRP，Berra运行RIP。Mantle的路由配置见示例11-17。

img574_2

图11-8 Ford运行IGRP，Berra运行RIP。Mantle正在进行路由重新分配

示例**11-17** **Mantel**同时运行**IGRP**和**RIP**，并在两个协议之间重新分配路由，其中使用缺省度量和**redistribution**命令指定的度量

img574_3

这里同时使用两种分配度量的方法是出于示范目的，在大部分情况下，如果重新分配方案和本例一样简单的话，可以使用一种方法。

注意，**Mantle**还被连接到一个末梢网络（192.168.10.0/24）。在本案例中，要求向**IGRP**域通告末梢网络，但是不能向**RIP**域通告。一种实现方法是仅在**IGRP**中添加适当的网络语句。然而，这样做会在末梢网络中造成不必要的**IGRP**广播。另一个实现方法是使用重新发配，具体配置见示例11-18。

示例**11-18** 路由器**Mantel**向**IGRP**重新分配直连末梢网络

img575_1

命令**redistribute connected** 将会重新分配所有直连网络。如果要向IGRP域和RIP域通告网络192.168.10.0/24，那么具体配置见示例11-19。

示例**11-19** 从路由器**Mantle**的配置可以看出，直连末梢网络被通告到**RIP**和**IGRP**，其中向**RIP**通告的路由度量值是在重新分配过程中指定的，而向**IGRP**通告的路由度量值由**default metric**命令指定

img575_2

11.2.2 案例研究：重新分配EIGRP和OSPF

图11-9的网络中，有一个OSPF域和两个EIGRP域。路由器Hodges运行OSPF进程1，Podres运行EIGRP进程1，Snider和Campanella运行EIGRP进程2。

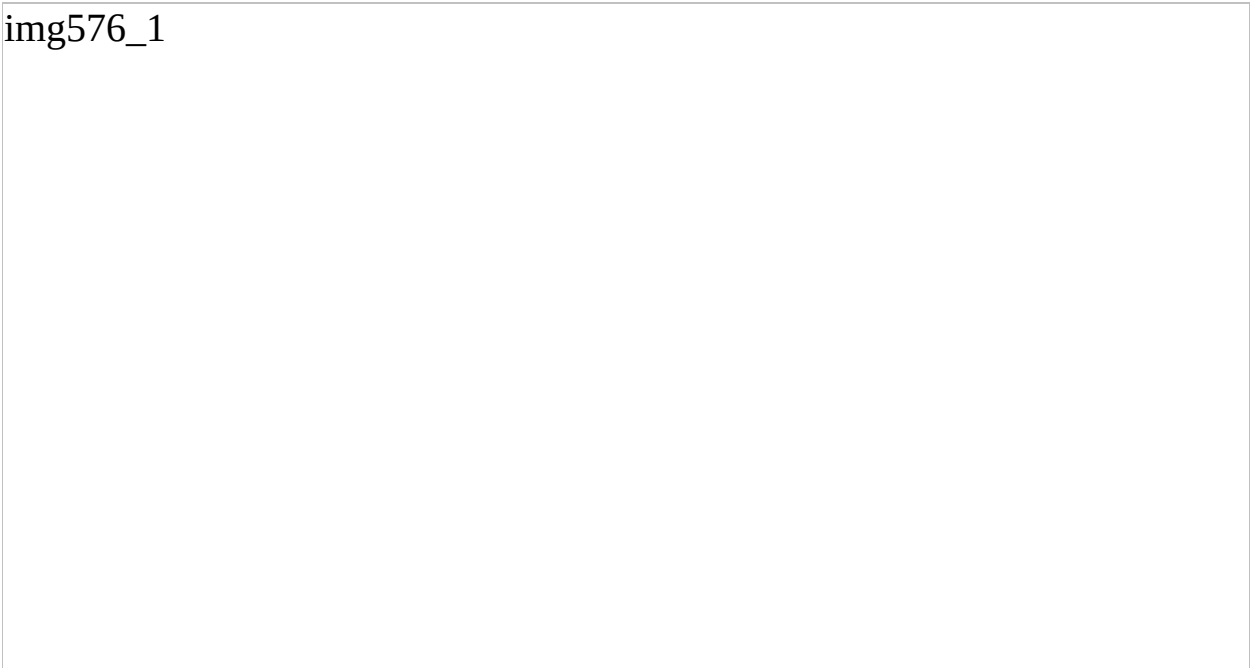
Robinson的配置见示例11-20。

示例11-20 Robinson向EIGRP1、EIGRP2和OSPF进程重新分配路由

img575_3



img576_1



img576_2




图11-9 Hodges运行OSPF，Podre运行EIGRP1，Snider和Campanella运行EIGRP2

注意，尽管在EIGRP进程之间必须配置重新分配，但是不需要配置度量。因为这些进程使用相同的度量，所以能够穿过重新分配边界准确地跟踪度量。示例11-21显示了Podres的路由表，重新分配的路由被标记为EIGRP外部路由。

示例11-21 图11-9中Podres的路由表

img576_3

img577_1

示例11-22显示了Hodges的路由表，这里存在一些问题。回忆一下第8章的内容，被重新分配到OSPF域的路由类型可以是类型1（E1）或类型2（E2）外部路由。在这里，被重新分配且标记为E2的路由仅是主网地址192.168.2.0/24、172.16.0.0和192.168.4.0/24。造成这种现象的原因是，在Robinson的配置语句中缺少关键字**subnets**。如果没有该关键字，那么被重新分配的地址仅包括那些在非OSPF进程内的主网地址。


示例**11-22** **Hodges**的路由表仅包括重新分配的主网路由，标记为**E2**

img577_2

修改后的Robinson配置见示例11-23，增加了关键字subnets。

示例11-23 当向**OSPF**重新分配路由时，为了使子网地址和主网地址一起被重新分配，**Robinson**使用关键字**subnets**来解决这个问题

img577_3



修改的结果使图11-9中的子网出现在Hodeges的路由表中（参见示例11-24）。

示例11-24 在关键字**subnets**被添加到**Robinson**的重新分配配置中后，**Hodegs**便可以知道所有子网信息

img578_1

缺省情况下，外部路由作为类型2路由被重新分配到OSPF。正如第8章所讨论的，E2路由仅包括路由的外部代价。如图11-10所示，当存在不止一条外部路由可以到达单一目标网络时，这一事实将显得十分重要。在图11-10的网络中，一台路由器正在重新分配代价为50，指向10.2.3.0/24的路由；另一台路由器也正在重新分配代价为100并且指向相同目标网络的另一条路由。如果这条路由被作为E2通告，那么在OSPF域内的链路代价将不会被计入。结果在OSPF域内的路由器将会选择路由1到达10.2.3.0/24。

img578_2

图11-10 如果去往10.2.3.0/24的路由被作为E2通告，那么路由1的代价将会是50，路由2的代价为100。如果该路由被作为E1通告，那么路由1的代价为150，路由2的代价为110

如果在图11-10中，指向10.2.3.0/24的路由被作为E1重新分配，那么在OSPF域内的链路代价将被计入重新分配代价。结果是，OSPF域内的路由器将选择路由2，代价为110（100 + 10）；而不是路由1，代价为150（100 + 50）。

图11-9中的路由器Robinson正在重新分配代价为50的EIGRP 1和代价为100的EIGRP2。示例11-24显示出，在Hodegs，指向EIGRP 1子网的路由代价仍然为50，而指向EIGRP2子网的路由代价为100。因为在Hodegs和Robinson之间的快速以太网链路代价没有被计入。

为了将路由作为E1重新分配到OSPF，可以在重新分配命令中添加关键字**metric-type 1**。在示例11-25的配置中，Robinson继续把EIGRP 1作为E2重新分配，但是把EIGRP 2作为E1重新分配。

示例11-25 经配置，**Robinson**把**EIGRP2**路由作为类型**1**外部路由向**OSPF**重新分配

img579_1

在重新配置Robinson之后，示例11-26给出了Hodegs的路由表。在EIGRP 1域内所有指向目标网络的路由代价仍旧为50，但是在EIGRP2域内指向目标网络的路由代价现在为101（重新分配代价加上Robinson和Hodegs之间快速以太网链路的缺省代价1）。

示例11-26 修改**Robinson**的配置以便把子网**192.168.4.0**和**172.16.0.0**作

为类型**1**外部路由通告

img579_2

img580_1

11.2.3 案例研究：重新分配和路由汇总

Cisco对EIGRP、OSPFv2、OSPFv3和IS-IS的实现都可以汇总被重新分配的路由。这个案例研究将分析在IPv4下EIGRP和OSPF的路由汇总；下面两个案例研究将分析IPv6下OSPFv3和IS-IS的路由汇总。

第一个要注意的事情是汇总要起作用，先决条件是IP子网地址已为汇总进行过规划。例如，在图11-9中OSPF域内，192.168.3.0的子网全部都被汇总地址192.168.3.0/25所包含。在EIGRP 1域内相同主网地址的所有子网也都被汇总地址192.168.3.128/25覆盖。如果子网192.168.3.0/27被连接到Podres，那么必须从汇总地址中将这个单一目标分离出来之后，才能进行通告。虽然通告单一目标几乎不会有什么不利影响，但是通告大量汇总地址范围之外的子网将会减少汇总带来的好处。

命令**summary-address** 为OSPF进程指定了一个汇总地址和掩码。任何在指定汇总地址范围内的更精确的地址都会被禁止。注意，此命令仅用在ASBR汇总外部路由；在ABR内部OSPF路由的汇总可以通过命令**area range** 实现，详见第8章。

在图11-9中路由器Robinson上，EIGRP1的子网在进入OSPF域时被汇总为192.168.3.128/25，EIGRP2的子网被汇总为172.16.0.0/16，具体配置见示例11-27。

示例11-27 Robinson对通告给OSPF的外部地址进行汇总

img580_2

比较示例11-28和示例11-26，在示例11-28中，Hodges的路由表包含指定的汇总地址，汇总地址范围以内的子网地址在重新分配点被禁止。注意，由于没有对192.168.4.0/24进行汇总配置，所以该主网地址的所有子网仍出现在路由表中。

对于EIGRP的汇总是指定接口的。也就是不在路由进程下指明汇总地址和掩码，而是在独立的接口下指明。这个系统提供了更大的灵活性，可以在同一进程的不同接口通告不同的汇总地址。命令**ip summary-address eigrp process-id** 指定了汇总地址、掩码和汇总所要通告的EIGRP进程。

在示例11-29的配置中，Robinson将向EIGRP 1通告汇总地址 192.168.3.0/25、172.16.0.0/16和 192.168.4.0/24。

示例11-28 Robinson 正在汇总 **192.168.3.128/25**和 **172.16.0.0/26**,因此在 **Hodges**的路由表中不会出现在此范围内的更精确的地址

img581_1

示例**11-29** **Robinson**对通告到**EIGRP1**的路由进行汇总

img581_2

img582_1

示例11-30给出了Podres的路由表。正如OSPF汇总一样，EIGRP的汇总禁止通告汇总范围以内的子网。但与OSPF不一样的是，Podres的路由表显示向EIGRP通告的汇总路由没有被标记为外部路由。

示例11-30 Podres的路由表显示了汇总路由192.168.3.0/25、192.168.4.0/24和192.172.16.0.0/16

img582_2

Robinson正在向Campanella通告EIGRP汇总路由192.168.3.0/24，同时向Snider通告192.168.0.0/16。示例11-31给出了Campanella的路由表，示例11-32给出了Snider的路由表。

示例11-31 在**Robinson**配置汇总之后的**Campanellade**的路由表

img582_3

示例**11-32** 在**Robinson**配置汇总之后的**Snider**的路由表

img583_1

回过头再看示例11-30，注意指向192.168.3.128/25的汇总路由，它可能会使你感到惊讶，因为汇总地址被通告到OSPF，而不是EIGRP。另外还要注意，这条路由被标记为外部路由，这表明它已经被重新分配到EIGRP域了。这里所发生的情况是，汇总路由被通告到OSPF，接着又从OSPF域被重新分配到EIGRP。所以在Podres出现了不期望的路由条目。这就是172.16.0.0/16和192.168.4.0/24作为外部路由出现在Campanella的原因。这些汇总地址先是通告给Podres的EIGRP1，接着又被重新分配给EIGRP2。Snider路由表中有指向172.16.0.0/16的路由，但没有指向192.168.4.0/24的路由，因为Snider有一条汇总地址为192.168.0.0/16的路由。

现在假设子网192.168.3.192/27变为不可访问。Podres将按照较精确的路由192.168.3.128/25转发去往该子网的数据包。数据包将被发送到OSPF域，你可能会认为在OSPF域中汇总路由192.168.3.128/25将导致数据包被送回Podres。

事实上，这种情形不会发生。**Robinson**的路由表（参见示例11-33）中有许多汇总路由条目都把Null0接口作为连接接口。空接口是一个不知道去往哪里的软件接口——路由到它的数据包将会被丢弃。除了某些特例外，[\[2\]](#)每当路由器产生一条汇总路由，路由器同时还会生成一条指向空接口的路由。如果**Robinson**接收到一个去往192.168.3.192/27的数据包并且该子网不再可达，那么路由器将转发数据包至空接口。路由环路将在这一跳被打断。

示例11-33 Robinson的路由表。由于路由器产生了许多汇总路由，因此相应地有许多连接到空接口的汇总路由条目。这样可以防止路由环路

img583_2

img584_1

指向空接口的汇总路由对于防止环路非常有用，关于空接口的使用详见第12章的内容。然而，重新分配不正确的路由选择信息是根本不允许发生的。假设Podres到Robinson不是1跳而是10跳，那么方向错误的消息要经过很长的路线之后才会被丢弃。这个例子证明了在互相进行重新分配时——也就是当两个路由选择协议互相向对方重新分配它们各自的路由时——需要仔细地控制路由通告。在这种情况下，使用路由过滤（详见第13章）或路由映射（详见第14章）是绝对必要的。

前面的情景还展示了为使用汇总而付出的代价。虽然路由表的大小被减少，节约了内存和处理器的循环周期，但路由的精度却降低了；随着网络变得更加复杂，细节的损失将会增加路由错误的可能性。

11.2.4 案例研究：重新分配OSPFv3和RIPng

图11-11是一个IPv6的网络。路由器Griffey是3个分离网络的连接点。两个网络运行OSPFv3，另一个运行RIPng。Griffey在路由选择进程之间重新分配路由。

img584_2




图11-11 IPv6通过OSPFv3和RIPng进行路由，其中一台路由器在路由选择进程之间进行路由的重新分配

路由器Griffey的配置见示例11-34。

示例11-34 Griffey上配置了IPv6、OSPFv3进程1、OSPFv3进程2和RIPng, Griffey在各进程之间进行路由的重新分配

img585_1

Griffey把来自RIPng进程Mariners的路由重新分配到进程ID为1的OSPFv3进程，路由的度量类型为外部类型1。而且，这些路由被添加了标记，值为4。在后面这些标记将被RIPng进程用来标识被通告的路由。Boone的IPv6路由表（参见示例11-35）给出了这些路由。

示例11-35 IPv6的路由表显示出来自RIP被重新分配到OSPF的路由，标记值为4，度量类型为外部类型1

img585_2

Griffey向OSPFv3进程（进程ID为2）通告RIPng路由，路由度量类型为缺省值（外部类型2），且没有标记值。在Garcia的路由表中可以看到这些被重新分配的路由（参见示例11-36）。

示例11-36 IPv6路由表给出了度量类型为缺省类型——外部类型2的被重新分配的路由

img585_3

img586_1

对IPv6来说，关键字subnets不能和OSPFv3一起使用。所有IPv6的路由选择协议都包括IPv6前缀，这些前缀通常包括地址和前缀长度。除非执行过滤机制限制前缀被通告，否则协议之间的路由重新分配会包括所有前

缀。第13章和14章将会讨论路由过滤机制。

Griffey也向RIPng重新分配从两个OSPFv3进程获知的路由。在被通告的路由中，来自OSPF1的路由度量值为5，来自OSPF2的路由度量值为10。示例11-37给出了Suzuki的IPv6路由表。

示例 11-37 从**RIPng**的路由表可以看出，被重新分配到**RIPng**的路由带有不同的度量值

img586_2

来自Boone的路由的度量值为6，来自Garcia的路由的度量值为11。

在Griffey上将要配置路由汇总。当RIPng重新分配来自其他协议的路由时，RIPng没有办法对这些通告来的路由进行汇总。但是在Griffey连接到Suzuki的以太网接口上，我们可以通过配置汇总从该接口通告出去的

RIPng前缀。当RIPng前缀被通告进入路由进程时，OSPFv3可以对它们进行汇总。Griffey具体的配置见示例11-38。

示例11-38 当RIPng路由从EthernetO/O被通告或被重新分配到OSPFv3时，Griffey可以对它们进行汇总

img587_1

在示例11-39中，汇总路由出现在新的路由表中，而更精确的路由被删

除了。

示例**11-39** 路由表中包括汇总路由，但是不再包含属于汇总范围的更精确的路由

img587_2

img588_1

注意，Griffey把汇总路由2001:db8:0:20::/61通告给Suzuki。这是为通告到OSPFv3的前缀建立的汇总路由。OSPF又把这个汇总路由通告给RIPng。RIPng域内的路由器仍然能够正确的路由更加精确的前缀。从Suzuki的IPv6路由表（参见示例11-40）可以看出，汇总2001:db8:0:20::/61所包含的前缀是直接连接到路由器的。所以路由器将会把流量转发给更精确的前缀，而不是给汇总前缀。让汇总重新被通告回RIPng域的不利之处表现为，如果数据包的目标地址在汇总范围内，但是这个目标实际在网络中并不存在，那么数据包在被丢弃之前会被转发给Griffey。我们可以在路由重新分配期间对路由进行过滤，以便被通告到RIPng的前缀仅是那些已知存在于OSPFv3域内的前缀。重新分配期间的路由过滤将在第13章和14章讨论。

示例1140 Suzuki的路由表中即包括汇总路由，也包括属于汇总范围的更精确的路由。路由器将尽可能的向更精确的路由转发流量

img588_2

img589_1

11.2.5 案例研究：重新分配IS-IS和RIP/RIPng

在图11-12的网络中，Aaron运行IS-IS, Williams为IPv4运行RIPv1,为IPv6运行RIPng,Mays正在重新分配路由。

img589_2

图11-12 路由器Mays正在向IS-IS重新分配RIP路由，同时向RIP重新分配IS-IS路由Mays的IS-IS和RIP配置见示例11-41。

示例11-41 路由器Mays的IS-IS、RIP和RIPng配置

img589_3

路由可能被作为内部或外部路由（缺省是内部）、第1层或第2层（缺省是第1层）路由向IS-IS重新分配。度量类型决定被重新分配路由的度量值基数。内部度量类型的度量值小于64，外部类型的度量值在64到128之间。如果度量类型是内部类型，那么被重新分配路由的初始度量将是度量值指定的数字（在本示例中度量值为0），如果没有指定度量值，那么则为0。在这个例子中，被重新分配的路由被添加进IS-IS数据库，表项的度量值为0。如果度量值为5，那么在Mays的IS-IS数据库中的表项度量值将是5。如果度量的外部类型基数从64开始且指定的度量值为5，那么Mays的IS-IS数据库中的表项度量值将是64+5或69。在所示例子中，RIP路由作为内部L2路由被重新分配，且使用缺省度量0。示例11-42给出了Aaron的IPv4路由表中被重新分配的路由，示例11-43给出了Aaron的IPv6路由表中被重新分配的路由。

示例11-42 **Araon**的路由表显示了被重新分配的**RIP**路由

img590_1

示例**11-43** **Aaron**的**IPv6**路由表给出了被重新分配的**RIP**路由

img590_2

Aaron的IPv6路由表显示，连接到Mays上的地址2001:da8:0:21::/64没有被通告到IS-IS。为了重新分配这些直连地址，在示例11-44中附加了命令redistribute connected，示例11-45给出了因此而产生的IPv6路由表。

示例11-44 因为在Mays上添加了**redistribute connected**，所以直连**IPv6**前缀被通告到**IS-IS**

img590_3

示例**11-45** **Aaron**的**IPv6**路由表给出了被重新分配的**RIP**路由和直连地址

img591_1

因为RIP对于IS-IS路由域来说是外部路由，把它们作为外部路由重新分配到路由域可以最好地反映这一点。详见示例11-46。

示例11-46 Mays把从RIP重新分配到IS-IS的路由配置为外部路由

img591_2

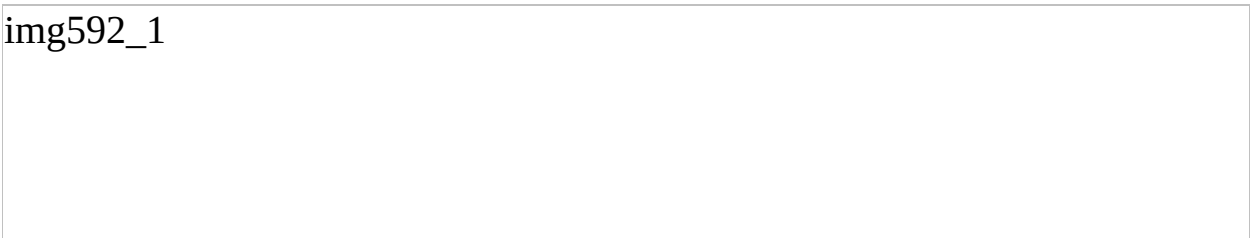
示例11-47给出了Aaron改变后的路由表。与示例11-42相比，惟一改变的是被重新分配的路由度量值变大了，并且大于64，这表明（在这个小型网络中）它们是外部路由。示例11-48给出了Mays的IS-IS数据库表项，其中被重新分配的路由表项被表示为IP-External和IPv6-Ext。

示例11-47 在路由被通告为外部路由后，去往**10.2.1.0/24**和**10.2.2.0/24**的路由度量值变为**138**

img591_3



img592_1



示例11-48 在Mays的IS-IS数据库中，被重新分配的路由被表示为**IP-External**和**IPv6-Ext**

img592_2

从图11-12还可以看出，RIP域内的IPv4和IPv6子网被汇总为10.2.0.0/16和

2001:db8:0:20::/62。命令**summary-address** 用于把IPv4路由汇总到IS-IS；OSPF也使用这个命令，而且还可以指定把汇总路由发送到哪一层。如果没有指定，那么地址将汇总到第2层。同OSPFv3一样，对于IPv6地址簇，将使用**summary-prefix** 汇总IPv6路由到IS-IS。在示例11-49的配置中，RIP的IPv4路由被作为L1重新分配和汇总，RIPng的IPv6路由被作为L2重新分配和汇总。

示例**11-49** 在**RIP**路由重新分配进入**IS-IS**后，**Mays**对它们进行汇总

img592_3

示例11-50给出了Aaron的IPv4路由表的汇总路由。示例11-51给出了Aaron的IPv6 路由表。像OSPF和EIGRP一样，这会导致在汇总范围内更精确的路由被抑制。

示例**11-50** **Aaron**的**IPv4**路由表中有一条指向**RIP**域子网的**L1**汇总路由

img593_1

示例**11-51** **Aaron**的**IPv6**路由表中有一条指向**RIP**域子网的**L2**汇总路由

img593_2

当IS-IS被重新分配到其他协议时，必须为重新分配的路由指定层数。到目前为止，在我们所举的例子中，重新分配到RIP的路由都被指定为L1和L2。

11.2.6 案例研究：重新分配静态路由

示例11-52给出了在图11-12中Williams的路由表。注意缺少了子网10.1.2.160/28和10.1.2.224/28。这些子网的掩码与配置在Mays接口E1上的24位掩码不一致，所以从该接口发送的RIP更新消息中没有包含这些路由。这个例子再一次说明了从无类别协议向有类别协议重新分配变长子网路由所存在的问题，这在本章开头曾讨论过。

示例11-52 没有向RIP域重新分配掩码不是24位的子网路由

img593_3

img594_1

解决这个问题的一种方案是使用24位地址10.1.2.0/24汇总两个28位子网。因为RIP没有这个汇总命令，所以实现该汇总的办法是配置一条指向汇总地址的静态路由，接着向RIP重新分配该路由。详细的配置见示例11-53。

示例11-53 在Mays上使用静态路由为IPv4的子网建立了一条汇总路由，并且把这条静态路由重新分配到RIP中

img594_2

示例11-54给出了Williams的路由表，其中包含了该汇总路由。

示例11-54 子网**10.1.2.160/28**和**10.1.2.224/28**被汇总为**10.1.2.0/24**

img594_3

正如第3章所讨论的，静态路由的一种变化形式是使用出站端口代替路由条目中的下一跳地址。这种静态路由也可以被重新分配，但是配置上稍微有一点不同。**Mays**的配置可参见示例11-55。

示例 11-55 在**Mays**上使用出站接口代替静态路由中的下一跳地址

img594_4

这里的静态路由指向Mays的接口E0，而不是指向下一跳地址10.1.4.1。由于在RIP的配置模式下不再使用命令**rdistribute static**，所以Williams的路由表看上去和示例11-54一样。

这个静态路由仍然会被重新分配的原因是当静态路由指向出站接口时，目标网络被认为是直接连接到路由器的（参见示例11-56）；又因为网络10.0.0.0的语句出现在RIP的配置中，所以RIP将通告10.0.0.0的直连子网。

示例11-56 Mays认为汇总地址**10.1.2.0/24**被直接连接到**Ethernet 0**

img595_1

假设Williams接收到数据包的目的地址是10.1.2.5，并且数据包匹配到汇总地址10.1.2.0/24，那么数据包将被转发给Mays。在Mays，目的地址不能匹配到更精确的子网，因而最终匹配到这条静态路由。Mays将在接口E0发送ARP请求，试图发现主机10.1.2.5（或者发现将发送一个代理ARP回应的路由器）。如果没有发现，那么路由器将不知道该如何处理此数据包。ICMP目标不可达信息将不会被发送给源点。

回忆一下，当使用汇总命令时，它们会在路由表中创建一个指向空接口的路由条目。

注意：空接口应该和静态汇总路由联合使用。

对于静态汇总路由，也应该做同样的工作（参见示例11-57）。

示例**11-57** **Mays**的静态汇总路由指向空接口

img595_2

现在，在路由器**Mays**，任何不能发现最精确匹配的目的地地址都会被路由到空接口后丢弃，同时目标不可达的**ICMP**消息将会被发送给源点。

11.3 展 望

本章讨论了在重新分配路由时会出现的几个问题。为了避免或纠正故障，除了最简单的重新分配方案之外，通常几乎在所有的方案中都包括对路由过滤和路由映射的使用，它们分别在第13章和第14章讨论。这些章节包括了更加复杂的重新分配方案以及如何进行故障诊断。首先，第12章将研究缺省路由——缺省路由被认为是汇总路由的最普遍的形式。

11.4 总结表：第11章命令总结

img596_1



11.5 复习题

1. 来自什么样信息源的路由可以被重新分配？
2. 管理距离的目的是什么？
3. 在重新分配时管理距离是如何导致故障的？
4. 从无类别路由选择协议向有类别路由选择协议重新分配是怎样导致故障的？
5. 哪一种IP的IGP可以使用缺省重新分配度量，为了重新分配工作正常，哪一种IGP必须配置度量？
6. 使用带关键字metric的redistribute命令和default-metric命令有什么区别？
7. 在重新分配OSPF时，关键字subnets的作用是什么？
8. 在汇总路由时空接口是如何起作用的？

11.6 配置练习

1. 在图11-13中，路由器A运行OSPFv2和OSPFv3，路由器C运行RIPv1和RIPng。为了使所有子网相互连通，请给出路由器B的配置。

img597_1

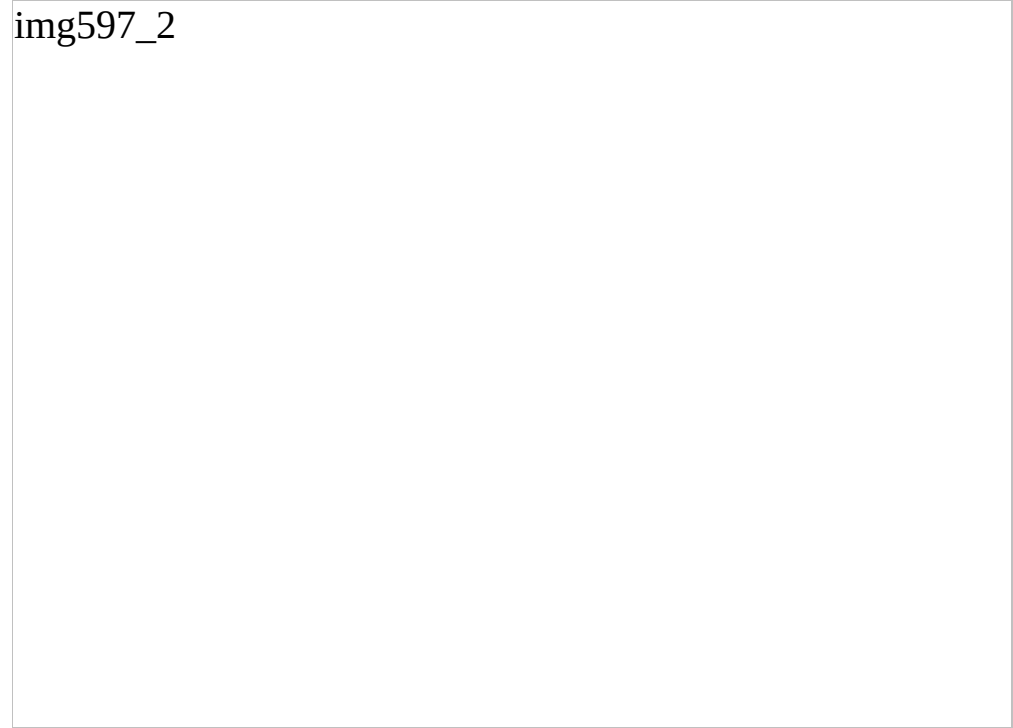
图11-13 配置练习1~3所使用的网络

2. 如图11-13所示，请尽可能在路由器B配置IPv6汇总路由。
3. 在图11-13中路由器A为IPv4和IPv6运行IS-IS。路由器C为IPv4运行EIGRP，为IPv6运行RIPng。在路由器B上，所有IS-IS的路由均为第1层。请在路由器B配置相互重新分配时尽可能使用汇总路由。EIGRP路由要求被作为外部路由通告到IS-IS域。

11.7 故障诊断练习

1. 在11.2.1小节中，下面给出图11-8中路由器Mantle的配置：

img597_2



由于路由可以被重新分配到IGRP，接着再次被重新分配到RIP，那么RIP域可以知道末梢网络192.168.10.0/24吗？

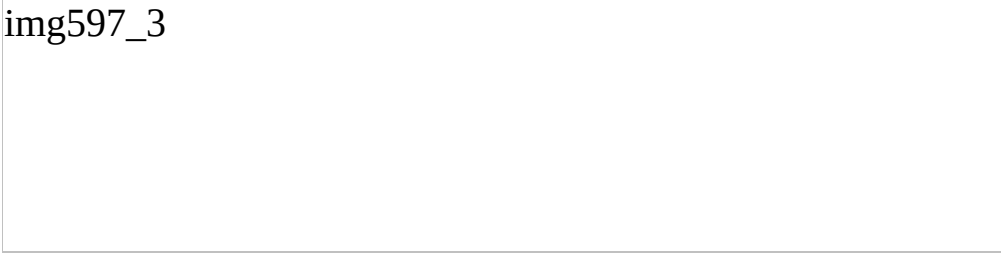
2. 在故障诊断练习1中，如果在IGRP的配置中取消命令`redistribute rip`，那么为了向RIP域进行通告，命令`redistribute connected`是否可以满足要求？

3. 在示例11-21中，为什么子网192.168.3.32/27没有被标记为EIGRP外部路由？

4. 在示例11-21中，有一条指向192.168.3.0的汇总路由，是什么导致产生了这一条目？

5. 根据以下配置，问区域1区的L1路由器是否可以知道汇总路由？

img597_3



[1] 在IP中，自治系统号相同的IGRP和EIGRP进程可以自动重新分配。

[2] 例如，OSPF内部区域汇总并不自动生成到空接口的汇总路由。它必须被静态配置。

本章包括以下主题：

- 缺省路由基本原理；
- 按需路由基本原理；
- 配置缺省路由和ODR。

第12章

缺省路由和按需路由选择

到目前为止，我们已经在几个章节中对路由汇总进行了讨论。汇总可以减小路由表的大小和路由通告内容，从而节省了网络资源。路由表越小、越简单，那么管理和故障诊断也越容易。

一个汇总地址可以表示几个或更多个更加精确的地址。例如，下面的4个子网可以用单一地址192.168.200.128/25来汇总。

192.168.200.128/27

192.168.200.160/27

192.168.200.192/27

192.168.200.224/27

当使用二进制方式查看地址时，我们可以看出汇总地址不太准确，因为汇总地址所包含的网络和子网位要比原地址少。因此，如果用粗略的方式表达，可以说向主机空间添加越多的0位，被使用的网络位就越少，那么可以汇总的地址就越多。按照这种想法，如果许多0位被添加到主机空间以至于没有剩余的网络位将会怎么样？换言之，如果汇总地址包括32个0位和前缀长度为0（0.0.0.0/0）又会怎样呢？这个地址将会汇总所有可能的IPv4地址。

0.0.0.0/0是IPv4的缺省地址，指向0.0.0.0/0的路由是缺省路由。[\[1\]](#)类似的，缺省IPv6地址::/0可以汇总所有IPv6地址。其他每个IP地址都比缺省地址更准确，所以当路由表中存在缺省路由时，如果不能寻找到一个更加匹配的路由，那么都会匹配到缺省路由上。

12.1 缺省路由基本原理

当将路由器与Internet相连时，缺省路由是非常有用的。使用了缺省路由，路由器仅需要知道它自己内部管理系统中的目标网络。缺省路由将把去往其他地址的数据包转发给Internet服务提供商。这样可以没有必要同服务提供商使用边界网关协议（BGP）去学习Internet路由表中的所有前缀——Internet路由表包含100000以上的前缀，并且可能很快会达到200000。在处理大型路由表时，拓扑变化所产生的影响远远大于对内存的需求。在大型网络中，拓扑频繁地发生变化，从而导致通告以及处理这些变化的系统活动明显增加。使用缺省路由可以有效地“隐藏”更精确路由的变化，使得具有缺省路由的网络更加稳定。

在单一自主系统中，缺省路由在更小的程度上还是有用的。在小型网络中，缺省路由可以减少对内存和CPU的使用，尽管随着路由数目的减少这种好处也会相应减弱。

缺省路由在星型（hub-and-spoke）拓扑结构中也非常有用，如图12-1所示。在这里，中心路由器包含指向每一个远程子网的静态路由。每当一个新的子网在线，中心路由器上就会被输入一条新的静态路由。虽然这一管理任务是微不足道的，但是要在每台分支路由器上添加路由可能会很耗时。如果在分支路由器上使用缺省路由，那么仅中心路由器需要有关每个子网的路由。当分支路由器收到去往未知目标网络的数据包时，它将把数据包转发至中心路由器，中心路由器接着将数据包发送到正确的目的地。

img600_1

图12-1 缺省路由大大地简化了星型拓扑网络的静态路由管理

在图12-1中，分支路由器更正确地应该被称为末梢路由器。末梢路由器到其他路由器仅存在一条连接。在这种设备上路由决策就变得非常简单：目标网络要么是路由器的直连网络（末梢网络）之一，要么经邻居路由器可达；而且如果这台邻居路由器是下一跳路由器的惟一选择，那么末梢路由器就不需要详细的路由表，一条缺省路由常常就足够了。

正如使用汇总路由一样，缺省路由也会造成路由细节的损失。例如在图12-1中的末梢路由器将不知道某个目标网络是否可达。所有去往未知目标网络的数据包都要被转发到中心路由器，然后确定网络是否可达。在网络中，很少会发生向不存在的地址发送数据包的情况。但如果万一发生，那么更好的设计选择是让末梢路由器运行路由协议，并从中心路由器获取路由以便尽快地确定未知网络。对于像图12-1这样的网络，你的设计选择或者是把未知目标网络的数据包转发给中心路由器，由它负责丢弃，或者在末梢路由器和中心路由器之间运行动态路由选择协议，并且末梢路由器就地丢弃去往未知目标网络的数据包，这两种方式哪种更经济。虽然运行动态路由选择协议所需要的资源和运作代价通常是很小的，但是缺省路由依然更可能是最佳选择。

图12-2给出了路由细节损失所引起的另一个问题。这些路由器形成了一个全国性的骨干网络，而且把大量本地网络连接到这些骨干路由器上。在洛杉矶（Los Angeles）骨干路由器上存在指向旧金山（San Francisco）和圣地亚哥（San Diego）的缺省路由。如果洛杉矶必须要向西雅图（Seattle）转发数据包，而它仅有两条缺省路由，那么它无法得

知经过旧金山才是最佳的路由。洛杉矶有可能会向圣地亚哥转发数据包，在这种情况下，数据包将使用部分非常昂贵的链路带宽，而且在数据包延迟到达目标网络之前将遭遇不必要的传播时延。虽然在骨干网中使用缺省路由是一个不好的设计决策，[\[2\]](#)但是却说明了使用缺省路由隐藏路由细节可能会导致不理想的路由选择。

img601_1



图12-2 如果洛杉矶仅知道指向旧金山和圣地亚哥通告的缺省路由，而不知道这两台路由器后面网络拓扑的更多细节，那么它将不能够有效地进行路由选择

12.2 按需路由基本原理

如图12-1所示，虽然在中心路由器上配置静态路由非常简单，但是许多网络管理员仍然不喜欢使用静态路由。困难不在于每当新的末梢网络在线时需要添加路由，而是在末梢网络或末梢路由器离线时忘记删除路由。从IOS 11.2起，Cisco开始向中心路由器提供另一种专有技术，叫做按需路由（On-Demand Routing，ODR）。

当末梢路由器仍然使用指向中心路由器的缺省路由时，中心路由器使用ODR可以自动地发现末梢网络。ODR仅传送地址前缀，即地址的网络号，而不是整个地址，因此路由器必须支持VLSM。由于在末梢路由器和中心路由器之间的链路上传输的路由信息非常少，所以节省了带宽。

ODR不是真正意义上的路由选择协议。它可以发现有关末梢网络的信息，但是ODR不能向末梢路由器提供任何路由选择信息。链路信息通过数据链路协议进行传输，因而从末梢路由器到中心路由器后不会做进一步的传输。然而，在后面的案例研究中将会讨论，ODR发现的路由可以被重新分配到动态路由选择协议中。

示例12-1给出了一个包含ODR表项的路由表，该表显示的管理距离为160，路由的度量值为1。因为ODR路由总是从中心路由器到末梢路由器，所以度量值（跳数）将永远不会超过1。路由还显示出支持VLSM。

示例12-1 这个路由表显示了几个ODR表项

img602_1



ODR路由的传输机制是Cisco发现协议（Cisco Discovery Protocol, CDP），CDP是一种专用的数据链路协议，它可以收集有关邻居网络设备[\[3\]](#)的信息。示例12-2给出了CDP所收集信息的类型。

示例12-2 CDP收集有关邻居Cisco网络设备的信息

img602_2



img603_1

CDP运行在任何支持子网访问协议（SNAP）的介质上，这意味着ODR还依赖于SNAP的支持。虽然在所有运行IOS 10.3或更高版本IOS的Cisco设备的所有接口上，CDP缺省是被启用的，但是从IOS11.2才开始支持ODR。配置案例研究部分将会显示ODR仅能配置在中心路由器上，为了中心路由器能发现末梢路由器所连接的网络，末梢路由器必须运行IOS 11.2或更高版本的操作系统。

12.3 配置缺省路由和ODR

缺省路由可以配置在每台需要缺省路由的路由器上，或者配置在向其对等路由器依次通告路由的路由器上。本节的案例研究将分析这两种方法。

回忆第5章讨论的有类别路由查询，路由器将首先匹配主网地址，然后匹配子网。如果子网不匹配，那么数据包将被丢弃。有类别路由查询是Cisco路由器IOS 11.3及后继版本的缺省模式。对于早期版本，可以通过命令**ip classless**把有类别查询变换到无类别（甚至对有类别路由选择协议）查询方式。

任何使用缺省路由的路由器必须执行无类别路由查询，图12-3解释了这样做的原因。在这个网络中，Memphis同Tanis、Giza使用动态路由选择协议，但是Memphis并不从Thebes接收路由。为了向BigNet路由数据包，Memphis有一条指向Thebes的缺省路由。如果Memphis接收到目的地址是192.168.1.50的数据包，并且它正在执行有类别路由查询，那么它将会首先匹配主网地址192.168.1.0，在路由表中存在该主网地址的几个子网。接着Memphis试图寻找有关子网192.168.1.48/28的路由，但是因为Memphis并不从Thebes接收路由，所以该子网不在路由表中，因此数据包会被丢弃。

如果Memphis配置了**ip classless**，那么它首先不会匹配主网络，而是为192.168.1.48/28寻找最精确的匹配。如果在路由表中没有发现，那么它将匹配到缺省路由，最终数据包被转发到Thebes。

img604_1



图12-3 **Memphis**使用缺省路由向**Thebes**转发数据包。如果**Memphis**使用有类别路由查询，子网**192.168.1.48/28**将不可达

12.3.1 案例研究：静态缺省路由

图12-3中Memphis的配置见示例12-3。

示例12-3 在**Memphis**上使用静态**IPv4**和**IPv6**路由建立缺省路由

img604_2

静态路由配置了缺省路由地址0.0.0.0和::/0，并且使用的掩码也是0.0.0.0（对IPv6来说前缀长度为0）。第一次配置缺省路由的人常犯的一个错误是使用全1掩码，而不是使用全0掩码，例如：

```
ip route 0.0.0.0 255.255.255.255 192.168.1.82
```

全1掩码将会设置一条指向0.0.0.0的主机路由，惟有那些目的地址为

0.0.0.0的数据包才能匹配到该地址。另一方面，全0掩码全部是由“不关心”位组成，它可以在任意位置匹配到任意位。本章开头曾讨论过缺省地址是汇总地址的一种极端形式，即每一个位都会被0汇总。这里缺省路由的掩码也是汇总掩码的一种极端形式。

Memphis缺省路由的下一跳地址在Thebes上。这个下一跳地址指的是最后可选网关或缺省路由器。示例12-4给出了Memphis的IPv4路由表。指向0.0.0.0的路由被标记为候选缺省，在表的开头指明了最后可选网关。示例12-5给出了IPv6的路由表。

示例12-4 Memphis的IPv4路由表，给出了缺省路由和最后可选网关

img605_1

示例12-5 **Memphis**的IPv6给出了指向缺省地址::<0的静态路由表项

img605_2

img606_1

通过向RIP重新分配静态路由，可以把缺省路由通告给剩余的RIP路由器。但Memphis不会向Tanis和Giza通告缺省路由，除非静态路由被重新分配到RIP协议[\[4\]](#)。在示例12-6中向Memphis配置中添加了IPv4和IPv6的重新分配命令。

示例12-6 向**Memphis**添加重新分配命令使得**RIP**通告静态缺省路由

img606_2

虽然OSPF和IS-IS不使用命令**redistribution** 通告缺省路由，但是它们也产生缺省路由，这可以在后继案例研究中看到。示例12-7和示例12-8分别给出了向RIP重新分配静态缺省路由后Tanis的IPv4和IPv6的路由表。

示例12-7 **Tanis**的**IPv4**路由表显示了缺省路由是通过**RIP**协议从**Memphis**那里学习到的

img606_3



示例12-8 Tanis的IPv6路由表显示了缺省路由是通过IPv6 RIP协议从Memphis那里学习到的

img606_4



img607_1

缺省路由对于连接无类别路由选择域也是非常有用的，在图12-4中，Chimu将一个RIP域和一个EIGRP域连接到一起。虽然在RIP域主网192.168.25.0的掩码是一致的，但是在EIGRP域对该主网进行了变长子网划分。此外，VLSM方式对进入RIP的汇总没有任何帮助。

img607_2

图12-4 缺省路由使**RIP**可以路由到变长子网化的**EIGRP**域

Chimu的配置见示例12-9。

示例12-9 **Chimu**把**RIP**路由重新分配到**EIGRP**，但除了缺省路由外所有**EIGRP**路由都没有被重新分配到**RIP**域

img607_3

img608_1

Chimu有一套来自EIGRP域的完整路由，但是Chimu没有将它们重新分配到RIP。相反，Chimu仅通告了一条缺省路由。RIP路由器将向Chimu转发所有去往未知网络的数据包，然后Chimu查找路由表，寻找一条到EIGRP域的最精确的路由。

Chimu的静态路由不是指向下一跳地址，而是指向空接口。如果转发给Chimu的数据包的目标属于一个不存在的子网，例如192.168.25.224/28，那么数据包不会被转发到EIGRP域，而是被丢弃。

12.3.2 案例研究：缺省网络命令

配置缺省路由的另一种方法是使用命令**ip default-network**。该命令指明了用作缺省网络的网络地址。这个网络可以是由静态路由指定的直连网络，也可以是通过动态路由选择协议发现的网络。开始介绍这个命令是和IGRP一起使用的，IGRP不用0.0.0.0作为缺省路由，而是把实际的网络标记为缺省网络。该命令仅用于IGRP、EIGRP和RIP。

命令**ip default-network**是一条全局命令，所以它会导致在支持这个命令的路由器上配置的所有路由协议都通告一条缺省路由。如果协议是IGRP和EIGRP，那么缺省路由就是命令参数指定的网络，如果是RIP，则是0.0.0.0。

在图12-5中，Athens的配置中使用了RIP和命令**ip default-network**，具体配置见示例12-10。

img608_2

图12-5 在Athens上使用命令**default-network**可以产生一个缺省网络通告

示例12-10 RIP使用命令**default-network**建立缺省路由

img608_3

如示例12-11所示，在Athens的路由表中网络10.0.0.0被标记为候选缺省路由，但是注意，没有指定最后可选网关，原因是Athens是到缺省网络的网关。即使在RIP配置中不存在有关网络10.0.0.0的语句（参见示例12-12），**ip default-network** 也将导致Athens通告一个缺省网络。当使用RIP时，Athens上配置的命令**ip default-network** 会使Athens把0.0.0.0作为缺省路由进行通告，而不是命令**ip default-network** 指定的网络。

示例12-11 在Athens的路由表中网络**10.0.0.0**被标记作为候选缺省路由

img609_1



示例12-12 **Sparta**的路由表显示出**Athens**正在通告一条缺省路由**0.0.0.0**，而且**Athens**是**Spartade**的最后可选网关

img609_2

和RIP一样，如果静态路由0.0.0.0被配置，那么EIGRP将向邻居通告一条缺省路由，同时还会重新分配静态路由。正如第7章所讨论的，EIGRP会把被重新分配的路由作为外部路由进行通告。

如果配置图12-5中的路由器运行EIGRP，并且使用命令**ip default-network**，那么Athen的配置见示例12-13。

示例12-13 命令**default-network**和**EIGRP**一起使用可以把一个网络标记为候选缺省路由

img610_1

命令**ip default-network** 保持不变，但是注意，在EIGRP的配置中添加了关于网络10.0.0.0的语句。因为EIGRP使用真实网络地址作为缺省网络，所以必须对该地址进行配置，详见示例12-14。比较示例12-12和示例12-14的路由表可以发现，RIP把指向0.0.0.0的路由标记为缺省路由，而EIGRP则把指向10.0.0.0/8的路由标记为缺省路由。因为Corinth从Aparta那里学习到缺省路由，所以Sparta是Corinth的最后可选网关。如果到Sparta的链路发生故障，那么Corinth将使用Argos作为最后可选网关。

示例12-14 EIGRP使用真实网络地址，而不是0.0.0.0，作为缺省网络。Corinth的路由表显示出网络10.0.0.0被标记为缺省网络

img610_2

注意在Athens的配置中，命令**ip default-network** 是一个全局命令，它不与特殊的路由选择协议相关联，也就是路由器上配置的任何路由选择协议都可以使用它，但前提是路由器要支持这条命令。如果路由器上配置了RIP和EIGRP，那么它们都会通告一条缺省路由，这可以在示例12-15中Conrinth的路由表中看到。

示例12-15 当Athens配置了RIP和EIGRP，并且使用命令**ip default-netowrk**，从Corinth的路由表可以看到有两条候选缺省路由

img610_3

img611_1

因为EIGRP的关联距离更小，所以来源于EIGRP的缺省网络成为最后可选网关。

使用这种方法通告缺省路由有一个固有的缺陷。如果路由器上配置了多个协议，例如RIP和EIGRP，使用命令**ip default-newwork**，那么就没有办法控制和限制由谁来通告缺省网络。在如图12-5中，如果Athens为BigNet运行EIGRP，而余下的网络运行RIP，配置**ip default-network**

的目的是向RIP通告缺省路由，同时Athens也向EIGRP通告缺省路由。这样做不仅会使从RIP域去往Bignet的流量的路由发生中断，而且还会影响Bignet内部的路由。

当向一个路由选择协议注入路由时，最好是选择一个可以提供最多控制的方法来最小化无目的路由的扩散。

12.3.3 案例研究：缺省信息始发命令

OSPF的ASBR和IS-IS域间路由器不能自动地向它们的路由选择域通告缺省路由，即使在缺省路由存在的时候也一样。例如，假设图12-5中的Athens配置了OSPF，并且设置了一条指向BigNet的缺省路由，Athens的配置见示例12-16。

示例12-16 Athens配置了OSPF，并且还有一条静态缺省路由

img611_2

示例12-17给出了Athens和Sparta的路由表。虽然静态路由使得在Athen上设置了最后可选网关，但是Sparta却不知道缺省路由。缺省路由必须以类型5的LSA方式被通告到OSPF域，这意味着Athens必须是一个ASBR。然而到目前为止，Athens的配置并没有告诉它执行这些功能。

示例12-17 在Athens的OSPF进程不能向OSPF域自动通告缺省路由

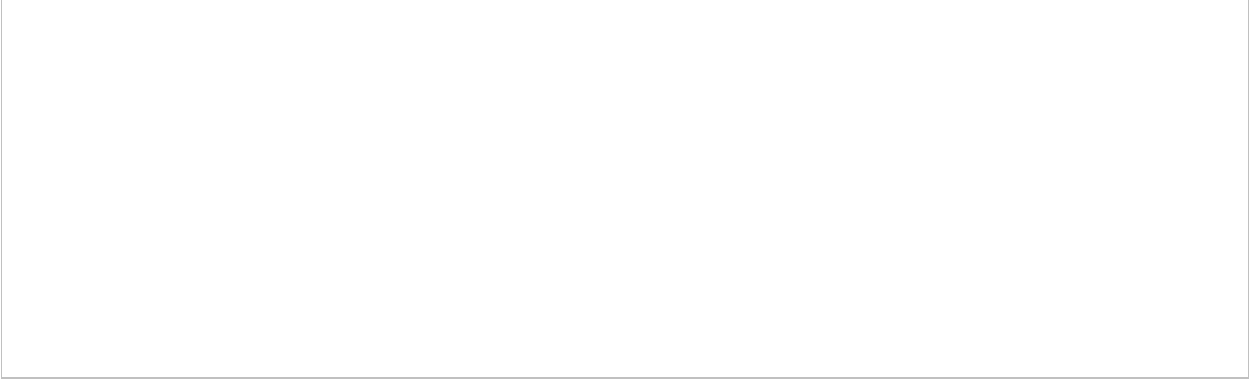
img611_3

img612_1

命令 **default-information originate** 是重新分配命令的一个特例，它将导致一条缺省路由被重新分配到OSPF或IS-IS。同 **redistribute** 一样，命令 **default-information originate** 通知OSPF路由器它成为一个ASBR或通知IS-IS路由器它成为一个域间路由器。而且还像 **redistribute** 一样指明被重新分配的缺省路由的度量，可以是OSPF外部度量类型，或者是IS-IS层级。为了向OSPF重新分配缺省路由，并且要求缺省路由的度量值为10，外部度量类型为E1，示例12-18给出了Athens的配置。

示例12-18 在Athens上命令 **default-information originate** 用于产生一条缺省路由


img612_2



示例12-19给出的缺省路由正在被重新分配到OSPF。而且在Sparta的OSPF数据库（参见示例12-20）中也可以观察到这个路由。

示例**12-19** 在**Athens**上配置**default-information originate**之后，缺省路由将被重新分配到**OSPF**域

img612_3




示例**12-20** 像**ASBR**通告的其他外部路由一样，缺省路由以类型**5**的**LSA**方式被通告

img613_1

命令 **default-information originate** 还可以向OSPF和IS-IS重新分配被其他路由选择进程发现的缺省路由。在示例12-21中，指向网络0.0.0.0的静态路由被删除，而Athens与BigNet中的一台路由器使用BGP通信。

示例12-21 配置Athens使用BGP获取路由，而不再使用静态路由

img613_2



现在Athens从它的BGP邻居路由器那里学习到指向0.0.0.0的路由，并且使用类型5的LSA向OSPF域通告该路由（参见示例12-22）。

示例12-22 在**BigNet**中使用**BGP**的路由器向**Athens**通告缺省路由

img613_3

缺省路由或汇总路由的好处是提供网络的稳定性，但是如果缺省路由自身不稳定会发生什么呢？例如在示例12-19中，假设通告给Athens的缺省路由在波动，也就是频繁地在可达与不可达之间变换。伴随着每一个变化，Athens都必须向OSPF域发送一条新的类型5的LSA，这个LSA将会被通告到所有非末梢区域。虽然这种泛洪和再泛洪对系统资源影响不大，但是，这不是网络管理员所期望的。解决办法是使用关键字**always**。[\[5\]](#)采用示例12-23的配置方法，即使路由表中没有缺省路由，Athens也总是会产生一条缺省路由。

示例12-23 即使路由表中没有缺省路由，**Athens**也将总是产生一条缺省路由

img614_1

使用这种配置方法，Athens将始终通告一条缺省路由到OSPF，不管它实际上是否有一条指向0.0.0.0的路由。如果在OSPF域内的一台路由器把Athens作为出口并转发了一个数据包，而Athens没有缺省路由，那么它将向源地址发送目标不可达的ICMP信息并且丢弃该数据包。

当OSPF域外仅有单一的缺省路由时，关键字**always** 可以安全地被使用。如果不止一个ASBR通告了缺省路由，那么缺省出口应该是动态的——即缺省路由的丢失将会被通告。如果一个ASBR在它没有缺省路由时却声明有缺省路由，那么数据包将会被转发到它那里，而不是被转发到合理的ASBR。

对于IPv6, **default-information originate** 的工作方式很类似。在图12-5中，IPv6由IS-IS进行路由。通告配置Athens将为IPv6产生一条缺省路由。

Athens的配置见示例12-24。


示例12-24 Athens为IS-IS协议产生一条IPv6的缺省路由

img614_2

在向IS-IS数据库输入缺省路由并向其他邻居通告缺省路由之前，Athens不需要从别的信息源那里获取缺省路由。其他路由器会把所有去往未知IPv6地址的数据包都转发给Athens。如果Athens也没有路由，那么它就丢弃这些数据包。示例12-25是关于Argos上的Athens的第2层IS-IS数据库表项；示例12-26是Argos的IPv6路由表。

示例12-25 IPv6缺省路由被添加到L2的IS-IS数据库中

img615_1



示例**12-26** **IPv6**缺省路由被作为**L2**的**IS-IS**添加到**IPv6**路由表中

img615_2

img616_1

12.3.4 案例研究：配置按需路由选择

使用命令**router odr** 可以启用ODR，不需要指明网络和其他参数。CDP缺省情况下是被启用的，仅在因某种原因被关闭的情况下才需要被启用。在路由器上启用CDP进程的命令是**cdp run** 。为了在特定接口上启用CDP，要使用命令**cdp enable** 。

图12-6给出了一个典型的星型拓扑结构。为了配置ODR，中心路由器必须配置命令**router odr** 。只要所有路由器都运行IOS 11.2或更高版本，并且连接介质支持SNAP（例如帧中继或PVC），ODR就可以运行，而且中心路由器将会学习到末梢网络。在末梢路由器上惟一需要配置的是指向中心路由器的静态缺省路由。

img616_2

图**12-6** 向这样的星型拓扑结构在帧中继网络中很普遍

ODR还可以被重新分配。在图12-6中，如果Baghdad需要向OSPF通告ODR发现的路由，可以使用示例12-27的配置。

示例**12-27** **ODR**发现的路由可以被重新分配到其他**IP**路由选择协议中

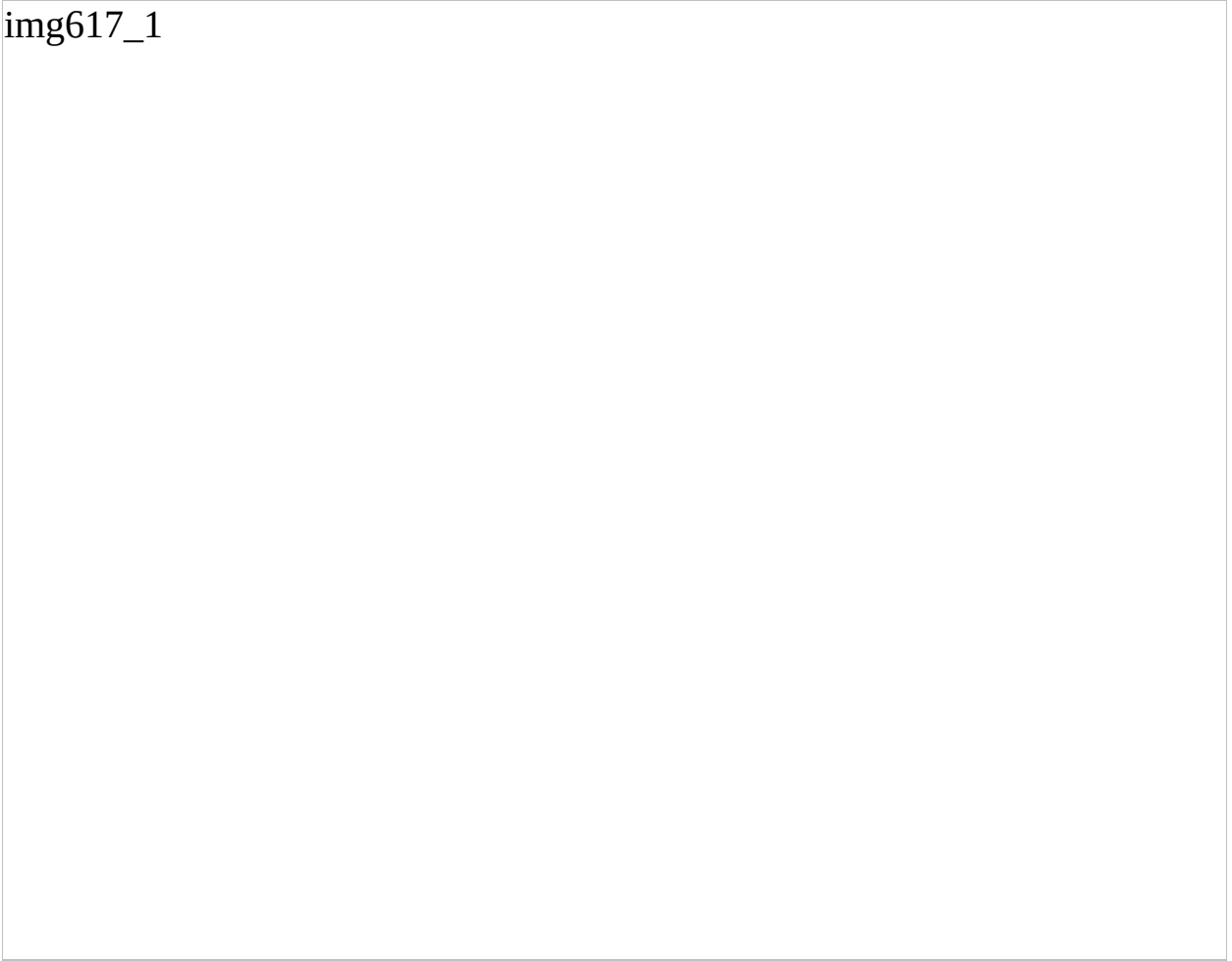
img616_3

12.4 展 望

在简单且无环路网络中，本章讨论的配置和故障诊断缺省路由是一种微不足道的任务。当拓扑结构更加复杂，特别是包含环路时，由缺省路由和重新分配所带来的潜在问题将会增加。第13章和第14章将讨论在复杂拓扑结构中控制路由行为的重要工具。

12.5 总结表：第12章命令总结

img617_1



12.6 复习题

1. 开放协议使用的IPv4缺省路由的目标地址是什么？
2. IPv6缺省路由的目标前缀/前缀长度是什么？
3. EIGRP如何通告和标识缺省路由？
4. 在运行EIGRP的路由器上可以使用指向0.0.0.0的静态路由作为缺省路由吗？
5. 什么是末梢路由器？什么是末梢网络？
6. 使用缺省路由代替完整的路由表的好处是什么？
7. 使用完整的路由表代替缺省路由的好处是什么？
8. 按需路由使用什么样的数据链路协议来发现路由？
9. 在使用ODR时，对IOS有什么限制？
10. 在使用ODR时，对介质有什么限制？

[1] 在所有开放式IP路由选择协议中均使用这个地址。Cisco的IGRP和EIGRP使用一个真实的网络地址，作为外部路由进行通告。

[2] 另一方面，让每台骨干路由器仅向它的本地网络通告一条缺省路由将会是一个非常好的设计选择，这可以限制本地路由表的大小。

[3] CDP不仅可以运行在路由器上，而且还可以运行在Cisco的交换机和接入服务器上。

[4] 在IOS12.0T之前的版本中，如果路由表中存在缺省路由，则不需要向RIP、IGRP和EIGRP重新分配静态路由，它们就会自动向邻居通告。

[5] 这个关键字仅在OSPF下可用，在IS-IS下不支持。

本章包括以下主题：

- 配置路由过滤器。

第13章

路由过滤

第11章中曾提到过在特殊路由器上，重新分配可能会在几种情况下导致不必要或不正确的路由。例如在图11-3及相关讨论中，一台或多台路由器选择了一条经过网络的非最佳路由，问题主要出在路由器更加信任IGRP，因为IGRP的管理距离比RIP要小。更加普遍的是，任何时间指向相同目标网络的路由都会被多台路由器重新分配到路由选择域，其中可能会存在错误的路由选择。在某些情况下，可能会发生路由选择环路和黑洞。

示例11-17给出了另一个关于不必要路由或意外路由的例子。在这个案例中，不仅汇总路由192.168.3.128/25被通告到OSPF，而且该路由还被重新分配到EIGRP域，然而EIGRP域即为被汇总子网的所在地。这种被通告路由沿错误方向穿过重新分配路由器的现象叫做路由回馈（route feedback）。

路由过滤可以使网络管理员对路由通告施加严格的控制。任何时刻路由器从一种协议向另一种协议重新分配路由，管理员可以使用路由过滤控制哪些路由被重新分配；同样的，路由器执行相互重新分配——在两个或多个路由选择协议之间相互共享路由——使用路由过滤器可以确保沿着惟一的方向通告路由。

图13-1给出了路由过滤器的另一种用途，在这里，一个路由选择域被分割为多个子域，每个子域包含多台路由器。

连接两个子域的路由器将对路由进行过滤，以便子域B中的路由器仅知道子域A中的部分路由。这种过滤可能是处于安全考虑，以便B域中的路由器仅知道已被授权的子网；或者仅仅是通过减少不必要的路由来维持B域内路由器的路由表和更新信息的大小。

此外，路由过滤器的另一个常见用途是建立路由防火墙。公司企业或政府机关常常需要被互连在一起，然而它们却处于独立的管理控制之下。如果你不能控制网络所有部分，那么你很容易受到错误配置的影响，

甚至恶意路由的攻击。如果在互连路由器上使用路由过滤，那么将确保路由器仅接收合法的路由。这种方法是一种安全的形式，但是在这种情况下，管制的是出站路由，而不是入站路由。

img620_1

图13-1 路由过滤器可以用于建立路由子域，以便仅向子域通告路由选择域内的部分地址

无论哪一种应用，路由过滤器都作为基本构建单元，被用于创建路由选择策略（**routing policy**）。路由选择策略是控制网络中数据包如何转发以及改变数据包缺省转发属性的一组规则。

外部的路由可以进入到路由表中，路由表中的路由也可以被通告出去，那么路由过滤器正是通过管制这些出入路由表的路由来工作的。路由过滤器对链路状态路由选择协议的影响和对距离矢量路由选择协议的影响稍微有点不同。运行距离矢量协议的路由器是基于自身路由表通告路由的，其结果是路由过滤器将会对路由器通告给其邻居路由器的路由产生影响。

另一方面，运行链路状态协议的路由器是基于自身链路状态数据库的信息来确定它们的路由，而不是基于被邻居路由器通告的路由条目。路由过滤器对链路状态的通告或链路状态数据库[\[1\]](#)没有影响。所以路由过滤器会对配置了过滤器的路由器的路由表产生影响，但不会对邻居路由

器的路由条目有任何影响。正因为这种特性，路由过滤器主要被用在进入链路状态域的重新分配点上，例如OSPF的ASBR（自主系统边界路由器），在那里路由过滤器可以控制那些进入或离开该域的路由。在链路状态域内，路由过滤器的效用是有限的。

13.1 配置路由过滤器

使用下面所给出的任意一种方法都可以实现路由过滤：

- 使用命令 **distribute-list** 过滤特定路由；
- 使用命令 **distance** 操作路由的管理距离。

13.1.1 案例研究：过滤特定路由

图13-2显示出一部分网络运行RIPv2和RIPng。Barkis经Traddles提供了到网络其余部分的连接。除了BigNet内700个明确的IPv4路由之外，Traddles还向Barkis通告了一条IPv4缺省路由。由于这条缺省路由，Barkis、Micawber、Peggotty和Heep不需要知道BigNet中700条以外的路由。因此，在Barkis上配置过滤器的目的是从Traddles仅接收缺省路由，并拒绝其他所有路由。Barkis的配置见示例13-1。

示例13-1 在Barkis上配置路由过滤器，允许缺省路由进入，拒绝所有其他地址

img621_1

img621_2

图13-2 在**Barkis**上，路由过滤器仅接收来自**Traddles**的缺省路由，并拒绝**BigNet**所有其他路由

该路由过滤器检查从接口S1入站的路由，其中S1是连接Traddles的接口。路由过滤器指定Barkis的RIP进程仅接收那些被访问列表1许可的路由，其中访问列表1指明仅允许0.0.0.0。[\[2\]](#)访问列表隐含地拒绝了所有其他路由。示例13-2给出了Barkis的路由表。

示例13-2 在来自**Traddles**的路由中，**0.0.0.0**是惟一被接受的

img621_3

IPv6的路由也可以按照相同的办法进行过滤。如示例13-3的配置，Barkis仅接受来自Traddles的缺省路由。

示例13-3 **Barkis**通过过滤**IPv6**路由仅接收缺省路由

img622_1

IPv4和IPv6配置惟一的不同是，IPv4的命令 **redistribute-list** 参照访问列表，而IPv6则参照一个前缀列表。前缀列表可以允许和禁止IPv6前缀。

示例13-4给出了Barkis的IPv6路由表。

示例**13-4** **::/0**是从**Traddles**那里接受的惟一的**IPv6 RIP**路由

img622_2

在示例13-5中，对Barkis的配置进行了改动，使其从2001:db8:0:103::/64

那里接受2001:db8:0:100::/64，而不再接受缺省IPv6路由。修改后的前缀列表仅接受这个范围的地址。

示例13-5 **Barkis**仅接受一段IPv6地址进入RIPng

img622_3

在**prefix-list** 命令中，**ge**和**le**可以指定IPv6前缀的范围，**ge**表示大于或等于，**le**表示小于或等于。**Barkis**的前缀列表指定了前缀2001:db8:0:100::/62，其中前62位相同，63或64位可以是任意值。这个范围包括2001:db8:0:100:/64、2001:db8:0:101:/64、2001:db8:0:102:/64和2001:db8:0:103:/64。前缀列表在附录B中将进一步讨论。

当然，沿串行链路通告的700条路由没想到会在链路的远端被全部丢弃，这对带宽而言，自然是一种浪费。因此更好的配置方法是将过滤器放在Traddles上，仅允许向Barkis通告缺省路由，相应的配置见示例13-6。

示例13-6 配置Traddle过滤出站的RIP IPv4路由

img622_4

img623_1

对于IPv6，Traddles上的相应配置见示例13-7。

示例**13-7** 配置**Traddle**过滤出站**IPv6 RIPng**路由

img623_2

这里的过滤器配置看上去与原来的配置差不多相同，但它是过滤出站路由，而不是进站路由。示例13-8显示出从Traddles沿串行链路被通告的路由中仅包括缺省路由。

示例**13-8** 在**Traddles**上的过滤器仅允许向**Barkis**通告缺省路由

img623_3

在这两种配置中，Barkis将向Micawber和Peggotty通告缺省路由，配置不会影响Barkis 向Traddles通告的路由。

当在链路状态协议（例如OSPF）下配置命令**distribute-list** 时，关键字**out** 不能与接口联合使用，[\[3\]](#)因为不像距离矢量协议，链路状态协议不从自身的路由表中通告路由，没有更新信息被过滤。所以命令**distribute-list 1 out Serial1** 在链路状态协议下是没有意义的。

在图13-3中，还连接了另一组路由器。这部分网络——ThemNet，在独立的管理控制之下，Creakle也一样。因为BigNet的管理员不能访问和控制ThemNet内的路由器，所以应使用路由过滤器将来自Creakle发往BigNet的错误路由信息的可能性减到最少。例如，ThemNet正在使用缺省路由（或许为了访问内部Internet连接）。如果这个缺省路由被通告到BigNet，那么它将导致数据包被误转到ThemNet内，从而产生黑洞。

img624_1

图13-3 网络ThemNet不在BigNet管理员的控制之下

为了与ThemNet进行通信，Heep仅允许必要的路由通过，相应的配置见示例13-9。

示例13-9 Heep的配置对来自和去往Creakle的RIP路由进行过滤

img624_2

分布列表1仅允许接受由访问列表1指定的，且来自Creakle的路由。分布列表阻挡了缺省路由和任何其他路由，这些路由可能会被不正确地插入到ThemNet路由表中。

分布列表2在适当的位置用于确保BigNet是一个正常的邻居；它阻挡了BigNet的缺省路由，否则该缺省路由将导致ThemNet内出现问题，但是它允许BigNet的所有其他路由通过。

13.1.2 案例研究：路由过滤和重新分配

路由器在任何时候执行相互重新分配时，路由回馈都可能会存在。例如，在图13-4中，来自RIP方的路由会被重新分配到OSPF，并且再从OSPF重新分配回到RIP。因此，使用路由过滤器控制路由通告的方向将

是一种明智的方法。

img625_1

图13-4 Cruncher正在向**OSPF**重新分配**RIP**路由，而且还向**RIP**重新分配**OSPF**路由。路由过滤器将被用于阻止路由回馈

图13-4中的Cruncher可以在几个接口上同时使用RIP和OSPF, Cruncher的配置见示例13-10。

示例13-10 Cruncher同时运行**OSPF**和**RIP**，并且配置了过滤器用来控制通告到每一种协议的路由

img625_2

在上面的配置中，访问列表逻辑允许某些路由但拒绝所有其他路由。该逻辑也可以被颠倒过来，拒绝某些路由但允许所有其他路由。关于Cruncher的配置改动见示例13-11。

示例13-11 修改Cruncher的访问控制，使其拒绝指定的路由但允许所

有其他路由

img626_1

第二个访问列表的作用同第一个访问列表相同。在这两种情况下，到OSPF域内目标网络的路由将不会从RIP向OSPF通告，同样到RIP域内目标网络的路由也不会从OSPF向RIP通告。然而，第二个访问列表的配置需要仔细地管理，以便最小化将来出现的问题。例如，一个新的地址被添加到RIP域，如果路由过滤器没有相应的做出修改，那么该路由将向OSPF通告并且还会被通告回来，因此可能会导致路由环路。这就需要在过滤器中添加一条拒绝语句来控制不发生环路。如果使用第二个访问列表替代第一个访问列表，并且向RIP域添加一个新地址，那么这个路由是不会被通告到OSPF的，除非访问列表被修改。这时就不会产生路由环路，而且RIP域内到该地址的路由也是正常的。因为在第二个访问列表的末尾存在**permit any**，所以对列表的配置不像管理这样方便。为了向访问列表添加新的条目，整个列表首先必须被删除以便在**permit any**之前可以放置新的条目。[\[4\]](#)

为了方便地进行汇总操作，我们对图13-4中的子网地址作了精心的分配，产生了小型的访问列表，但却牺牲了精确性。对路由的控制越精确，意味着访问列表越大、越精确，这是以增加管理的注意力为代价的。

在重新分配点部署路由过滤器的另一种方法是借助路由进程进行过滤，而不是接口。例如，示例13-12的配置仅允许重新分配图13-4中的某些IPv4路由。

示例13-12 **Cruncher**在路由进程上过滤路由

img626_2

在OSPF配置下的路由过滤器允许OSPF通告RIP协议发现的路由，但这些路由必须是访问表10许可的路由。同样的，在RIP配置下的路由过滤器允许RIP通告OSPF 25发现的路由，但这些路由必须是访问列表20许可的路由。在两种情况下，路由过滤器对其他协议发现的路由没有影

响。例如，如果OSPF重新分配RIP和EIGRP的路由，前面的分布列表将不会应用到EIGRP发现的路由上。

同样的配置也可以应用在IPv6前缀上。对于RIPng和OSPFv3，分配列表会参照前缀列表，而不是访问列表，其中前缀列表被用来允许和禁止IPv6前缀，而不是IPv4地址。

当通过进程进行过滤时，仅允许使用关键字**out**。在OSPF下使用**distribute-list 10 in rip**是没有意义的，因为路由已经通过RIP进入到路由表中了，OSPF要么通告它（out），要么不通告。

注意，虽然通过路由选择协议进行过滤对于指定那些将要被重新分配的路由是很有用处的，但是它并不是防止路由回馈的好办法。例如，考虑一下在图13-4中Cruncher的配置（参见示例13-13）。

示例13-13 Cruncher的配置对**OSPF**和**RIP**进程通告的路由进行过滤，但是不过滤哪些已经被添加到路由表中的路由

img627_1

假设一条来自RIP域的路由，如172.16.190.0/24，被重新分配到OSPF域，然后又被通告回到Cruncher，原因是路由器配置错误。虽然在RIP配置下的分布列表将会阻止路由被通告回RIP域，但是它却不能阻止路由以OSPF域发生的路由的身份进入Cruncher的路由表。事实上，过滤器认为路由已经通过OSPF进入路由表，因为OSPF的管理距离低于RIP。Cruncher将会优选OSPF路由，因此Cruncher把去往172.16.190.0的流量路由到OSPF的路由器，而不是RIP路由器。所以为了阻止路由回馈，必须在路由进入路由表之前，在路由进站时进行过滤。

13.1.3 案例研究：协议迁移

命令 **distance** 可以为路由指定管理距离，这些路由是从一个特殊的IPv4或IPv6路由协议那里学习到的。命令在使用时不带任何可选参数。在最初考虑时，该操作看上去不像路由过滤功能，但是当运行多个路由选择协议时就不同了，这时将会基于路由的管理距离来确定是否接受或拒绝路由。

在图13-5中，网络运行RIP，并且计划将路由选择协议转换为EIGRP。有好几种方法能完成这样的路由迁移。一种方法是在每一台路由器上关闭老的协议，然后打开新的协议。虽然这种方法对类似于图13-5的小型网络来说是合适的，但在更大的网络中，这种停工其是不切实际的。

img628_1

图13-5 这些运行**RIP**的路由器将会被转为运行**EIGRP**

另一种选择是不删除旧协议的同时添加新的协议。如果新协议的缺省管理距离小于旧协议，那么每台路由器将选择新协议通告的路由。随着路由器转换完毕，网络将收敛到新的协议上。在整个网络收敛到新的协议之后，则可以从所有路由器上删除旧的协议。

参照表11-1，RIP的缺省管理距离是120，EIGRP的缺省管理距离为90。除了RIP之外，如果EIGRP被添加到每台路由器，那么路由器将随着邻居路由器开始使用EIGRP的同时也开始选择使用EIGRP路由。当所有RIP路由从路由表中消失时，网络将收敛到EIGRP。RIP进程接着会从路由器上被删除。

这种方法的问题是，在重新配置的时候可能会存在路由环路和黑洞。在图13-5中，大约几分钟就可以完成对5台路由器的重新配置和转换，所

以这里不会像大型网络一样关注环路问题。

对这种双协议方法的改进是使用命令**distance**，以确保禁止新协议的路由，一直到所有路由器为转换做好准备。这个过程中第一步是在所有路由器上降低RIP的管理距离，具体配置见示例13-14。

示例13-14 对图13-5网络中的每台路由器进行重新配置，减小**RIP**的管理距离

img628_2

注意，管理距离仅与单台路由器的路由进程相关。当RIP仍然是惟一正在运行的协议时，管理距离的改变不会影响到路由。

下一步是重新访问路由器，向每一台路由器添加EIGRP进程，EIGRP的配置见示例13-15。

示例13-15 为图13-5中的每台路由器添加**EIGRP**配置

img628_3

由于EIGRP的缺省管理距离是90，因此RIP的路由被优先选择（参见示例13-16）。因为没有路由器选择EIGRP路由，所以在配置期间不必确定停工时间表。这种方式使网络管理员在转换之前有时间重新检查每一台路由器的新配置。

示例**13-16** 由于为**RIP**路由分配的管理距离为**70**，所以优先选择的是**RIP**路由，而不是**EIGRP**路由


img629_1

协议转换过程的下一步是再一次访问每台路由器，将**RIP**的距离改回到

120。这一步要计划停工时间。因为新的RIP更新路由的管理距离被指派为120，所以管理距离为70的RIP路由将会超时（参见示例13-17）。经过210s之后，宣布RIP路由失效（参见示例13-18），最终EIGRP的路由将被优先选择（参见示例13-19）。

示例13-17 在RIP的管理距离被改回到120之后，管理距离为70的路由开始老化。这里所有RIP路由老化时间都已超过2min

img629_2



示例**13-18** 经过**3.5min**，**RIP**路由将被宣布失效

img630_1

示例13-19 缺省管理距离为90的EIGRP路由代替了RIP路由

img630_2

虽然使用这种方法仍然有可能产生路由环路和黑洞，但是因为仅需要修改管理距离，所以转换速度会更快，人为的错误也越小。

这种方法的另一优点是，万一出现问题可以很容易地停止切换工作。在所有路由器中RIP进程仍然处于合适的位置，退回到RIP需要做的所有工作就是将管理距离改回70。一旦新的EIGRP被测试并证明是稳定后，可以从所有路由器上删除RIP进程而不会有进一步的服务中断。

在使用双协议方法之前需要考虑一件事情，即同时在每台路由器上运行两个协议可能会影响路由器的内存用量和处理速度。如果内存利用率、处理利用率或两者的平均值超过50%或60%，那么在提交转换工作之前应该进行仔细的实验室测试和仿真，以确保路由器可以处理这些额外的负载。如果路由器不能胜任，那么在配置新协议之前需要更加复杂的过程来删除旧协议，这可能是惟一的选择。

与上面过程不同之处在于，这里是增加新协议的管理距离，而不是降低旧协议的管理距离，接着在转换时再降低新协议的管理距离。但是，要确保在输入任何网络命令之前输入命令**distance**，以便不会用新协议缺省的管理距离激活新协议。

回顾一下表11-1，注意EIGRP有两个管理距离：对内部路由为90，对外部路由为170。因此，对EIGRP来说，命令**distance** 还有些不同。例如，用提高EIGRP的管理距离代替降低RIP的管理距离，相应配置见示例13-20。

示例13-20 在图13-5中的所有路由器上提高EIGRP的内部管理距离，而不是降低RIP的管理距离

img631_1

添加关键字**eigrp** 以说明指定的EIGRP管理距离。内部EIGRP路由的管理距离被改为130，而外部路由的管理距离仍为170。

使用双协议迁移到新的路由选择协议还需注意的一点是：确认你已经理解两种协议的行为。例如，某些协议（如EIGRP）不会老化自身的路由条目。因此，如果要取代EIGRP，需要增加额外的一步，就是在改变完管理距离之后使用命令**clear ip route ***清除路由表。

13.1.4 案例研究：多个重新分配点


图13-6给出的网络非常类似于图11-3给出的网络。回忆一下第11章中关于多个重新分配点问题的讨论，由于管理距离导致路由器选择了不满意路由的问题。在某些情况下，还会导致路由环路和黑洞。例如，在Bumble的路由表（参见示例13-21）中，指向网络192.168.6.0的路由的下一跳路由器是Blathers，而不是Monks。

示例13-21 使用Bumble的路由可以经Blathers（192.168.3.2）到达192.168.6.0，这里包括两条串行链路和一个令牌环

img631_2

img632_1

img632_2

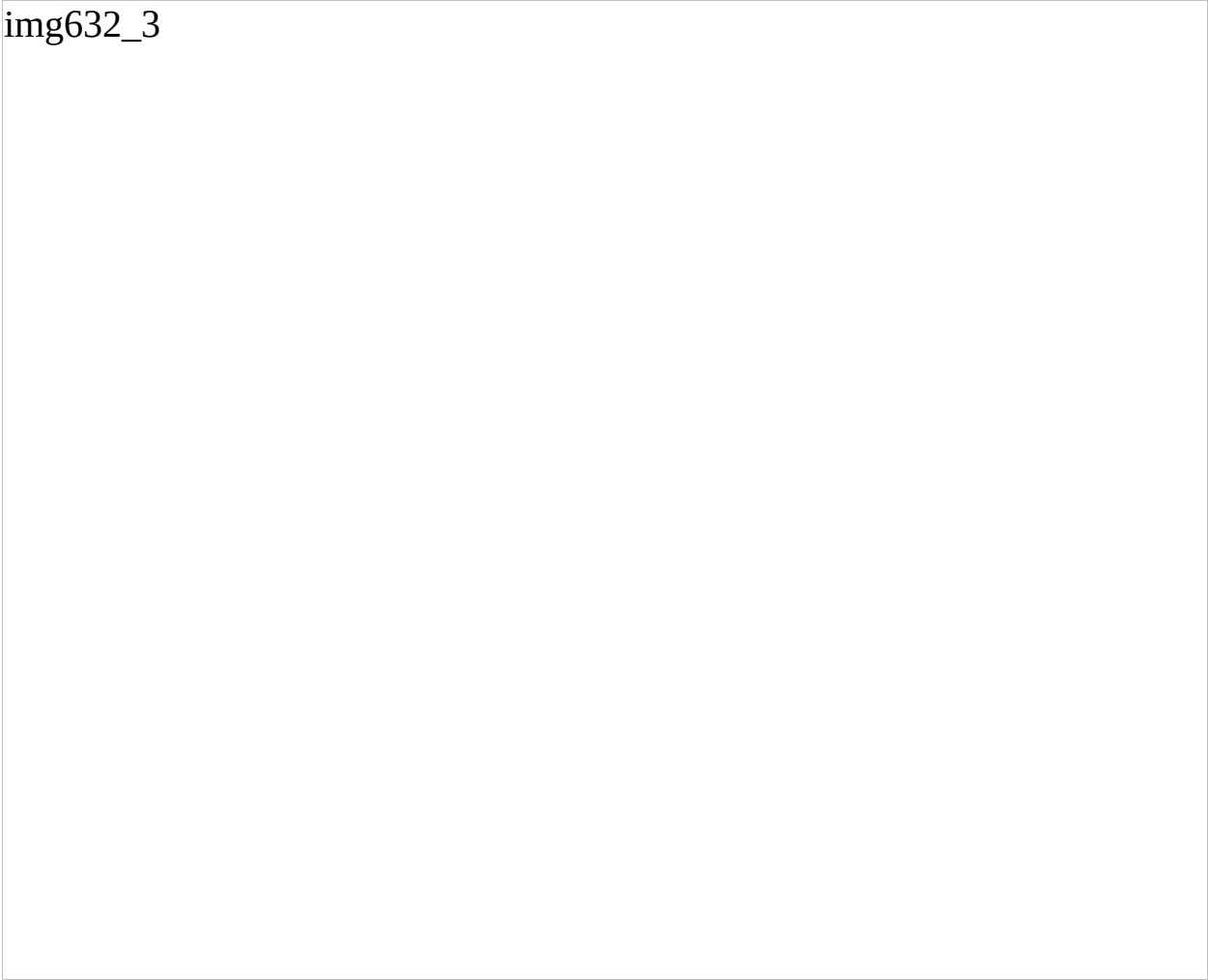


图**13-6** 当在多个点进行互相重新分配时，管理距离会导致非最佳路径选择、路由环路和黑洞

解决这个问题的一种办法是，在重新分配点使用命令**distribute-list** 来控制路由源点。Bumble和Grimwig的配置分别见示例13-22和示例13-23。


示例**13-22** **Bumble**配置使**OSPF**进程仅接受**OSPF**域的地址，以及**RIP**进程仅接受**RIP**域的地址

img632_3




示例**13-23** **Grimwig**配置使**OSPF**进程仅接受**OSPF**域的地址，以及**RIP**进程仅接受**RIP**域的地址

img632_4



img633_1



在上面两个配置中，访问列表1仅允许OSPF接受OSPF域内的地址，访问列表2仅允许RTP接受RIP域内的网络。在配置了路由过滤器之后，示例13-24给出了Bumble的路由表。

示例13-24 在配置路由过滤之后，**Bumble**将使用最佳路由到达网络**192.168.6.0**

img633_2

使用这种配置方法的问题是，消除了多个重新分配点内在的冗余性。在示例13-25中，**Bumble**的以太网链路被断开。由于在OSPF中过滤掉指向RIP网络的路由，因而所有路由现在都不可达。


示例13-25 当**Bumble**的以太网链路发生故障后，**RIP**网络变得不可达。路由过滤器可阻止**OSPF**向路由表中输入替代的路由

img633_3

对IPv4来说，一种更好的方法是使用命令**distance** 的两种形式来设置首选路由。[\[5\]](#) Bumble和Grimwig的配置见示例13-26和示例13-27。

示例13-26 Bumble使用距离配置命令修改IP路由的管理距离

img634_1



示例13-27 **Grimwig**使用距离配置命令修改**IP**路由的管理距离

img634_2

在这两个配置中，第一个**distance**命令设置了OSPF和RIP的管理距离为130。第二个**distance**命令根据被指定的通告路由器和参考访问列表来设定一个不同的管理距离。例如，Grimwig的RIP进程为由Monks（192.168.6.1）通告且被访问列表2许可的路由指定的管理距离为120，所有其他路由的管理距离为130。注意，这里与通告路由器地址一起使用的是反码。

在OSPF的配置中会存在更多的问题。通告路由器的地址不必是下一跳路由器的接口地址，而是产生LSA的路由器的ID，其中路由就是根据LSA进行计算的。因此对命令**distance**来说，在OSPF配置下的地址和反码是0.0.0.0 255.255.255.255，它们可以指定任意路由器。[\[6\]](#)访问列表1许可的OSPF路由的管理距离被指派为110，所有其他路由的管理距离为

130。

结果如示例13-28所示，第一个路由表显示Grimwig经过Duff到达OSPF域内的所有网络，Grimwig经Monks到达RIP域内的所有网络。OSPF正常的路由管理距离为110，RIP管理距离为120。接着，Grimwig的以太网链路发生故障。第二个路由表显示所有网络均经过Duff可达。指向RIP域内网络的路由的管理距离为130。当以太网链路恢复正常后，来自Monks且管理距离为120的RIP通告将取代替管理距离为130的OSPF通告。

示例13-28 Grimwig到Monks的以太网链路发生故障前后Grimwig的路由表

img635_1

在图13-6中，如果其中一条串行链路发生故障，那么将会发生相反的事情。穿过RIP域将可以到达OSPF域内的网络，而且管理距离再次是130（参见示例13-29）。然而，不像OSPF可以快速地收敛，RIP要花几分钟的时间才能收敛。这种慢收敛是由于在Monks处RIP的水平分割所导致的。Monks将不会向Bumble和Grimwig通告OSPF路由，直到这两台路由器停止通告相同的路由并且现存的路由超时为止。

解决问题的办法是，关闭Monks两个以太网接口上的水平分割功能，这可以使用命令**no ip split-horizon** 完成。虽然关闭水平分割缩短了收敛时间，但同时也失去了环路保护功能，不过还是值得的。在Bumble和Grimwig上基于管理距离的路由过滤可以阻止所有多跳环路，而且在两台路由器的相同以太网接口上的水平分割功能还可以打断单跳环路。

示例13-29 到Duff串行链路发生故障之前和之后的Grimwig的路由表

img635_2

img636_1

还有另一种在互相重新分配路由的路由选择协议之间帮助过滤地址的方法，就是路由标记（tag）。当路由被重新分配到某个路由选择协议时，这些路由可以被标记。标记是管理员设置的一个数字，可以惟一地标识重新分配到该路由器的一组路由集合。这个标记可以在另一台路由器上用于在重新分配回原来的路由选择协议时过滤掉这些路由。标记可以使用路由映射进行设置，这将在第14章中进一步讲述。

13.1.5 案例研究：使用管理距离设置路由器优先权

在图13-6中，假设策略规定把Grimwig作为到OSPF域的主路由器，仅当Grimwig不可达时才选择经过Bumble的路由。策略实施前，Monks通过在Grimwig和Bumble之间执行

等价负载均衡到达OSPF的网络（参见示例13-30）。

示例13-30 Monks把从Bumble和Grimwig到OSPF域内所有网络都看成是等距离的

img636_2

通过降低来自Grimwig的路由的管理距离，可以使得Monks优先选择Grimwig，相应配置见示例13-31。

示例13-31 Monks通告配置降低所有来自Grimwig的路由管理距离

img637_1

在这里，命令**distance** 没有参考访问列表。所有Grimwig（192.168.6.2）通告的路由的管理距离都将被指定为100。所有其他路由（来自Bumble）都被指定为RIP的缺省管理距离120。因此Grimwig的路由被优先选择。

示例13-32给出了结果。第一个路由表显示了Monks选择Grimwig进行路由。当到Grimwig的连接发生故障时，路由表结尾显示Monks切换到Bumble（192.168.2.2）。

示例13-32 到Grimwig的以太网链路发生故障之前和之后的Monks的路由表


img637_2

13.2 展望

对控制网络的行为来说，路由过滤器是一个非常有用的工具。在大型网络中，路由过滤器几乎是不可缺少的。但是路由过滤器的所有效用仅限于允许或不接受路由。第14章将介绍另一个强大的工具——路由映射，路由映射不仅可以标识路由，还可以主动地修改路由。

13.3 总结表：第13章命令总结

img638_1



13.4 配置练习

1. 在图13-7中路由器A的配置见示例13-33。请在路由器A上配置路由过滤器，禁止除路由器E之外所有路由器知道子网172.16.12.0/24。

示例13-33 图13-7中路由器A的配置

img638_2

2. 图13-7中，在路由器A上配置路由过滤器，阻止路由器D学习到子网172.16.10.0/24。
3. 图13-7中，在路由器A上配置路由过滤器，仅允许向RIP域通告子网172.16.2.0/24、172.16.8.0/24和172.16.9.0/24。
4. 图13-7中，在路由器A上配置路由过滤器，禁止路由器B学习到RIP域内的任何子网。
5. 路由器A、B和C上配置了OSPFv3，路由器A、D和E配置了RIPng。路由器B上连接了IPv6前缀2001:db8:0:1::/64、2001:db8:0:2::/64和2001:db8:0:3::/64。路由器E连接了2001:db8:0:a::/64、2001:db8:0:b::/64和

2001:db8:0:c::/64。请在图13-7中路由器A上配置路由过滤器禁止把2001:da8:0:a::/64和2001:db8:0:b::/64通告给路由器D。

img639_1




图13-7 配置练习1~4中的网络

6. 表13-1给出了图13-8中所有路由器的接口地址。路由器A和路由器B运行EIGRP，路由器E和路由器F运行IS-IS。路由器C和路由器D正在重新分配。为路由器C和路由器D配置命令**disntance**，来阻止路由环路和路由回馈，但允许存在冗余路由。

img639_2

表13-1 图13-8中所有路由器的接口地址

7. 在图13-8中，使用命令**distance** 配置路由器D，使它仅接受来自路由器A的EIGRP路由。如果到路由器A的链路发生故障，那么路由器D将不接受来自路由器B的路由，虽然路由器D仍旧向路由器B通告路由。

8. 删除练习7中添加到路由器D的配置，配置图13-8中的路由器C，使它通过路由器A可以到达所有目标网络，包括IS-IS域内的所有子网。仅在路由器A发生故障时，路由器C才选择经过路由器E和路由器F进行路由。



图13-8 配置练习6～8的网络

13.5 故障诊断练习

1. 路由器的配置见示例13-34。

示例**13-34** 路由器试图滤掉缺省路由

img640_2

上面配置的目的是阻止进入接口E5/1的缺省路由，而允许进入该接口的所有其他路由。但是，在接口E5/1路由器却没有接受任何路由，错误在哪里？

2. 在图13-6中，Grimwig的配置见示例13-35。

示例**13-35** 故障诊断练习2中**Grimwig**的配置

img640_3

img641_1

该配置对Grimwig的路由有什么影响？

3. 在图13-9中，路由器运行OSPF，路由器B的配置见示例13-36。

img641_2

图13-9 故障诊断练习3和4中的网络

示例13-36 图13-9中路由器B的OSPF配置

img641_3

上面配置的目的是禁止路由器B和路由器C具有关于网络172.17.0.0的路由表项。看上去该计划在路由器B上正在实施，但是路由器C还是具有关于172.17.0.0的路由表项，为什么？

4. 在图13-9中，路由器运行RIP，路由器B的配置见示例13-37。

示例13-37 路由器B的RIP配置

img641_4

上面配置的目的是向路由器A仅通告网络172.22.0.0，向路由器C仅通告网络172.18.0.0。但是，在路由器A和路由器C的路由表中均没有RIP路由表项。错误在哪里？

[\[1\]](#) 请记住，链路状态协议的基本要求是，一个区域内的所有路由器必须

具有一致的链路状态数据库。如果路由过滤器阻挡了一些LSA，那么将违反上面的要求。

[2] 注意没有给出反码。访问列表的缺省反码是0.0.0.0，这对于本配置是正确的。

[3] 关键字out可以与一种路由选择协议联合使用，这将在下一个案例研究中讨论。

[4] 在某些IOS版本中，使用序列号可以把访问列表条目添加到指定的位置，使得访问列表条目非常容易管理。

[5] IPv6路由协议也支持使用**distance** 命令修改管理距离；但在IPv6的命令格式中仅有一个选项可以改变距离值。目前还没有选项用于指定邻居地址或前缀列表。

[6] 相同的“任意”地址也可以和RIP一起使用，而使用一个特殊地址完全是为了示范目的。

本章包括以下主题：

- 路由映射的基本用途；
- 配置路由映射。

第14章

路由映射

路由映射与访问列表十分相似，它们都包含匹配确定数据包细节的准则、许可及拒绝这些数据包的操作。但是路由映射不像访问列表，它可以向匹配准则中加入设置准则，设置准则可以按照指定的方式真正地对数据包和路由信息进行修改，而且路由映射还有更多的选项用来匹配给定的数据包。总而言之，在你的网络中，路由映射是一个定制路由策略的强大工具。

14.1 路由映射的基本用途

路由映射可以用于路由重新分配和策略路由；虽然在前面几章对重新分配进行了广泛地讨论，但本章介绍的主题是策略路由。策略路由在大规模边界网关协议（BGP）的运行中，是一个必不可少的工具。对于使用路由映射实现BGP路由策略已超出了本书的范围，但在“TCP/IP路由技术”第二卷中包括了这个内容（CCIE专业开发）（1-57870-089-2）。

策略路由（policy route）只不过是复杂的静态路由。策略路由可以基于数据包源地址包头中的其他域向指定的下一跳路由器转发数据包，而静态路由是基于数据包的目的地址向指定的下一跳路由器转发数据包。策略路由还可以链接到扩展IP访问列表，使路由器可以基于协议类型和端口号进行路由选择。同静态路由一样，策略路由只会影响那些配置了策略路由的路由器。

图14-1给出了一个典型的策略路由应用的例子。AbnetNet经路由器Dogpath连接到两个Internet服务提供商。AbnerNet的公司策略规定一些用户的Internet流量经ISP1发送，而其他用户的Internet流量要经ISP2发送。如果其中一个ISP不可达，那么流经该提供商的流量将被倒换到另一个提供商。在Dogpatch上的策略路由会根据本地策略对Intenet流量进行分配。流量的分配可以基于子网、特殊的用户甚至用户应用。

img644_1

图14-1 策略路由允许来自AbnerNetde的流量被路由到两个Internet服务提供商中的一个，流量的分配可以基于诸如源地址、源/目的地址组

合、数据包大小或应用层端口等参数

图14-2给出了策略路由的另一种用法。右边的一个系统监测来自行星**Mongo**的入侵军队，而另一个系统保存了过去的**Dilbert**连环漫画。我们可以配置策略路由使得从**Mongo**系统到**Flash_G**的关键流量经过**FDDI**（光纤分布式数据接口）链路，而优先级较低的**Dilbert**流量经过**56kbit/s**链路。反之亦然。

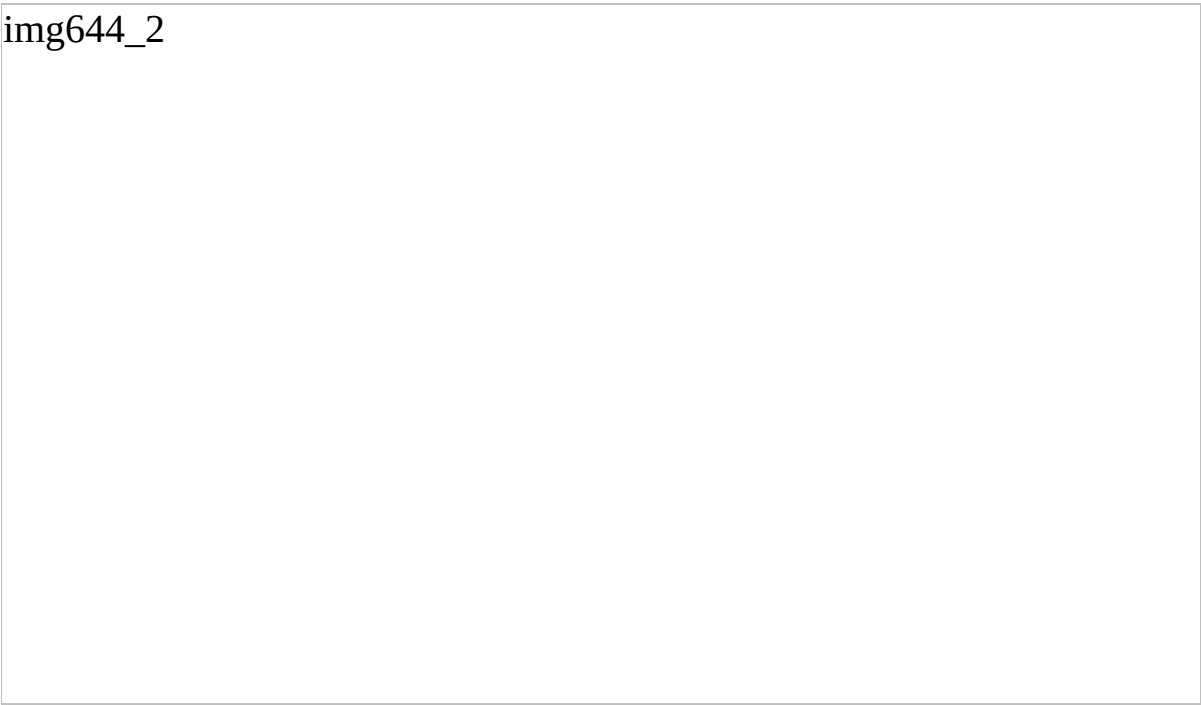


图14-2 策略路由允许来自**Mongo**系统优先级较高的流量使用**FDDI**链路，而来自**Dilbert**系统优先级较低的流量使用**56kbit/s**链路

表14-1和表14-2给出了与重新分配一起使用的命令**match** 和**set** ， 表14-3和表14-4给出了与策略路由一起使用的命令**match** 和**set** 。

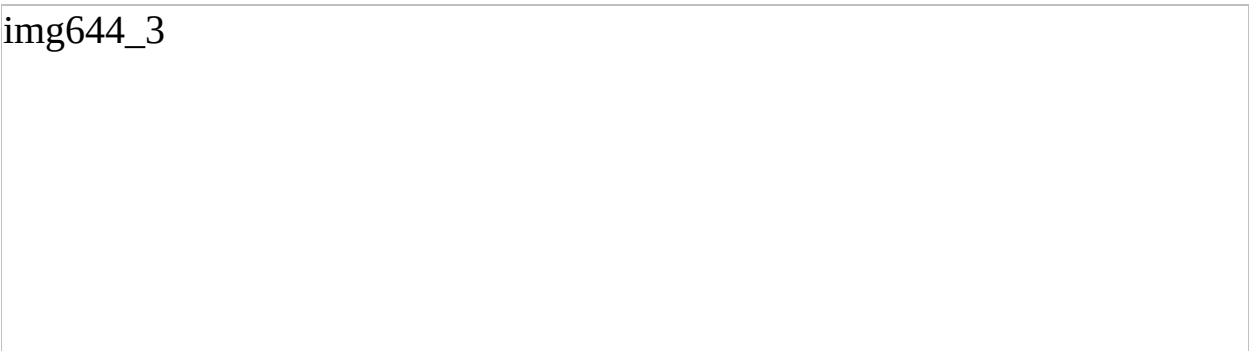


表14-1 match命令可以与重新分配一起使用

img645_1

表14-2 set命令可以与重新分配一起使用

img645_2

表14-3 match命令可以与策略路由一起使用

img645_3

表14-4 **set**命令可以与策略路由一起使用

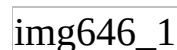
img645_4

14.2 配置路由映射

与访问列表一样（参见附录B），路由映射本身不会对任何东西产生影响，它们必须被某些命令所调用。这些命令最可能是策略路由选择命令或者是重新分配命令。策略路由选择将数据包发送到路由映射，而重新分配是将路由发送到路由映射。本节的案例研究将给出在重新分配和策略路由选择中路由映射的用法。

路由映射是通过名字来标识的。例如，在示例14-1中的路由映射被命名为Hagar。

示例14-1 配置中定义了名为**Hagar**的路由映射

The image is a rectangular placeholder with a thin black border. Inside the rectangle, the text "img646_1" is written in a black, sans-serif font, positioned in the upper-left corner. The rest of the rectangle is empty.

每条路由映射语句都包含“许可”、“拒绝”操作及一个序列号。这个路由映射给出了一个许可操作和序列号10。这些设置都是缺省的——也就是在配置路由映射时如果没有指定操作或序列号，那么路由映射的操作和序列号缺省值就是序列号10。

序列号允许说明和编辑多个语句，考虑示例14-2中的配置步骤。

示例14-2 修改路由映射**Hagar**

img646_2

这里，向路由映射Hagar添加了第二组和第三组路由映射语句，其中每组语句都包含自己的**match** 和**set** 设置语句。注意，第一次配置的序列号是20，第二次配置的序列号是15。如示例14-3所示，最终的配置中，尽管语句15是在后面被输入的，[\[1\]](#)但IOS还是把语句15放在了语句20的前面。

示例14-3 IOS按照顺序放置命令

img646_3

有了序列号以后，路由器还允许删除个别语句。例如下面的语句：

Linus (config) #no route-map Hagar 15

这里删除了语句15，而其他语句被完整无缺地保留下来（参见示例14-4）。

示例14-4 删除与序列号15相关的匹配/设置语句后的路由映射Hagar

img646_4

编辑路由映射时必须谨慎。在这个例子中，如果输入了**no route-map Hagar**，而没有指明序列号，那么整个路由映射将会被删除。同样的，如果在添加**match** 和**set** 语句时没有指定序列号，那么它们仅会修改语句10。 [\[2\]](#)

路由映射语句对数据包和路由的匹配是按序进行的。如果匹配成功，那么将执行所有**set** 语句及许可或拒绝操作。如同使用访问列表一样，当匹配发生时，处理马上停止，指定的操作将被执行；此后路由或数据包不再传递给后继语句。下面考虑示例14-5的路由映射。

示例14-5 路由映射Slugg。

img647_1

如果路由不能匹配到语句10，那么它将被传递到语句20。如果在语句20匹配发生，那么将执行设置命令，并且路由被允许。匹配成功的路由将不会被传递给语句30。

拒绝操作的行为依赖于路由映射是用于策略路由选择还是重新分配。如果用于重新分配，那么路由将不会被重新分配。如果用于策略路由选择，则不对数据包进行策略路由，但是数据包会被传递给常规路由选择进行转发。

与访问列表一样，如果数据包或路由没有匹配到任何一个语句，那么对路由映射来说必须执行一个缺省操作。在每个路由映射的结尾都隐含了一个拒绝操作。经过重新分配路由映射的路由若没有发生匹配，则不会被重新分配。而经过策略路由映射而没有发生匹配的数据包将会按常规路由选择进程转发。

如果在路由映射语句中没有配置**match** 语句，那么缺省操作是匹配所有数据包和路由。

在示例14-6中，每个路由映射语句可能有多个**match** 和**set** 语句。

示例14-6 Garfield包含多个与序列号10相关联的匹配和设置语句

img647_2

在这个案例研究中若要执行**set** 语句，每个**match** 语句中都必须发生匹配。

14.2.1 案例研究：策略路由选择

使用命令**ip policy route-map** 可以定义策略路由选择。该命令配置在接口且仅会对入站 数据包有影响。

在图14-3中，假设在Linux上实现了一个策略，使来自172.16.6.0/24的流量被转发到Lucy，而把来自172.16.7.0/24的流量转发到Pigpen。示例14-7给出了Linux的配置。

img648_1




图14-3 在**Linus**上配置策略路由，使某些数据包经过**Lucy**，而其他数据包经过**Pigpen**

示例14-7 在**Linus**上的策略路由配置

img648_2

S0上的策略路由选择命令把进站数据包发送给路由映射Sally。路由映射Sally的语句10使用访问列表1标识来自子网172.16.6.0/24的源地址。如果匹配成功，数据包将被转发到Lucy，其中数据包的下一跳接口地址是172.16.4.2。如果匹配不成功，数据包被发送给语句15。该语句使用访问列表2匹配来自子网172.16.7.0/24的源地址。如果匹配成功，数据包被转发给Pigpen（172.16.4.3）。任何没有匹配到语句15的数据包，例如来自子网172.16.8.0/24的数据包，将会被正常地路由。示例14-8给出了策略路由的结果。 [\[3\]](#)

示例14-8 配置在Linux接口S0上的策略路由把来自子网**172.16.6.0/24**的数据包路由到**Lucy（172.16.4.2）**，把来自子网**172.16.7.0/24**的数据包路由到**Pigpen（172.16.4.3）**。来自子网**172.16.8.0/24**的数据包因没有匹配到策略路由，所以被正常地路由（在**Lucy**和**Pigpen**之间进行负载均衡）

img649_1

假设Lucy的以太网接口发生故障。Linus将试图把来自172.16.6.0的数据包转发到Lucy的IP地址。因为在路由映射中数据包一旦匹配成功，即使指定的下一跳接口失效，也不会再次进行匹配，而且数据包也不能按常规被路由。在转发数据包之前为了强制Linus验证下一跳地址是否可用，可以使用命令**set ip next-hop verify-availability**。Linus将搜索CDP邻居表来验证下一跳地址是否在列表中；如果不在，则策略路由被拒绝，数据包将按常规被转发。示例14-9给出了当Lucy的以太网接口失效时**debug ip policy** 和**debug arp** 的输出结果。可以看出，数据包匹配成功并按照策略进行路由，但因路由器发出ARP请求后没有受到响应，所以数据包被丢弃。

示例14-10给出了在向IP的策略配置中添加命令**set ip next-hop verify-availability** 后命令**debug ip policy** 的输出。这时Lucy的以太网接口依然失效，数据包匹配成功，因为下一跳地址不在Linus的CDP表中，所以策略被拒绝，数据包最终按照常规方式被路由。

示例**14-9** 在**Linux**上的**debug ip policy**和**debug arp**显示：即使策略指定的下一跳不可用，路由器依然根据配置的策略对数据包进行路由

img649_2



img650_1

示例**14-10** **Linux**上的**debug ip**显示：在配置了**set ip next-hop verify-availavbility**后，因为下一跳不在**Linux**邻居表中，策略被拒绝，因而数据包按照常规方式被路由

img650_2

当Lucy的以太网接口恢复正常后，示例14-11给出了**debug ip policy** 的输出，可以看出数据包按照策略成功地被路由。

示例**14-11** 在配置了**set ip next-hop verify-availavbility**的**Linux**上使用命令**debug ip policy**，从结果可以看出：当对下一跳的验证成功后，数据包即可以被策略路由

img650_3

当仅按照源地址进行策略路由选择时可以使用标准IP访问列表。如果需要借助源地址和目标地址进行路由，那么要使用扩展访问列表。如果需要把从任意子网到主机172.16.1.1的数据包转发到Lucy，其次把从主机172.16.7.1到172.16.1.2的数据包转发给Pigpen，其他所有数据包被正常地路由，那么示例14-12给出的配置可以实现这个要求。


示例14-12 策略路由映射可以参考扩展IP访问列表对指定的源目地址对进行匹配

img650_4

这里再次使用路由映射Sally，现在除了**match** 语句参照访问列表101和102外，其他没有变化。产生的结果见示例14-13。

示例14-13 从主机**172.16.7.1**到主机**172.16.1.1**的数据包被路由映射**Sally**中的语句**10**匹配成功，因而被转发给**Lucy**。从相同主机到主机**172.16.1.2**的数据包则被转发到**Pigpen**。从子网**172.16.7.0/24**上的另一个地址到主机**172.16.1.2**的数据包没有被**Sally**匹配到，因而被正常地路由

img651_1



其次，假设你的策略规定：把从子网172.16.1.0/24上的服务器发出的FTP流量转发到Lucy，而把从相同服务器发出的Telnet流量转发到Pigpen。这个计划使大容量FTP流量和突发、活跃的Telnet在Schroeder处被分离到两条串行链路上。示例14-14给出了Schroeder的相应配置。

示例14-14 Schroeder的策略路由配置用于转发FTP和Telnet流量

img651_2

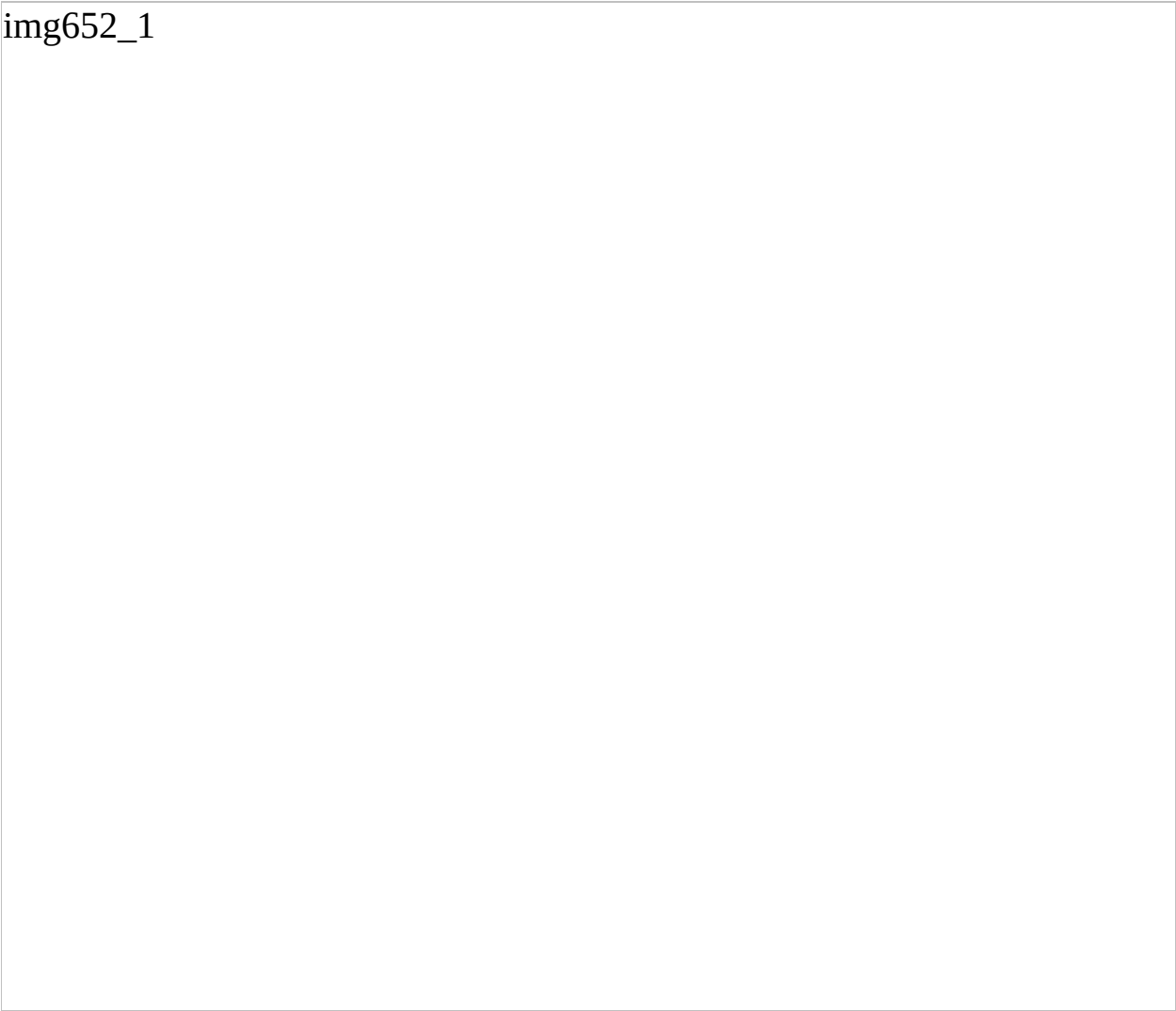
访问列表105和106不仅检查源地址和目的地址，而且还检查源端口。在示例14-15中，使用**debug ip packet**和选项**detail**，可以观察到Schroeder所转发的数据包类型。访问列表10对所要显示的数据包作出了限制，仅允许显示从172.16.1.1到172.16.6.1的数据包。

示例14-15 FTP数据包（TCP端口20和21）被转发到Lucy，而源地址和目的地地址相同的Telnet数据包（TCP端口23）则被转发到Pigpen。回应应答数据包（ICMP类型0）在策略路由中没有找到匹配语句，将被正常地路由

img651_3



img652_1



分割批量和交互式流量的目的是使小数据包特性的交互式流量不受大数据包特性的批量流量影响而增大时延。最后一个例子也是这样的。在本例中该方法的缺点是，如果需要隔离的流量类型很多，那么通过目标端口标识流量会使访问列表变得惊人的庞大。

如果策略的目标是将小数据包从大数据包中分离出来，那么可以对数据包长度进行匹配，相应配置见示例14-16。

示例14-16 Schroeder的策略路由配置将基于数据包长度转发流量


img652_2

在这里，**match length** 语句指明了数据包长度的最小值和最大值。路由映射的语句20使所有长度在1000~1600字节之间的数据包经过串行链路到达Lucy。语句30使所有长度大于400字节的数据包经过串行链路到达Pigpen。长度在400~1000字节之间的数据包则被正常地路由。

示例14-17给出了新路由映射的结果。从172.16.1.2到172.16.6.1的FTP、Telnet和回应应答数据包现在将根据它们的大小被路由，而不是按照数据包的地址和端口。

示例14-17 长度大于等于**1000**字节的数据包被路由到**Lucy**，而长度小于等于**400**字节的数据包被路由到**Pigpen**。所有长度在**400~1000**字节之间的数据包将被正常地路由

img652_3



img653_1

到目前为止，所示例的策略路由都是对从一个特殊接口进入路由器的数据包产生影响，但是路由器自己产生的数据包又会怎样？使用命令**ip local policy route-map** 可以使这些数据包也按策略进行路由。与**ip policy route-map** 不同，**ip policy route-map** 被配置在接口上，而命令**ip local policy route-map** 则是被全局地配置在路由器上。

为了对Schroeder产生的数据包应用前面示范的策略，示例14-18给出了

相应的配置。

示例14-18 针对Schroeder产生数据包的策略路由配置

img653_2

这里特别让人感兴趣的是语句10，该语句没有**set** 语句，而仅是允许访问列表120匹配到的数据包。访问列表120依次允许OSPF数据包和所有到子网172.16.1.0/24的数据包。如果没有访问列表的第一行，那么由Schroeder产出的数据包和到子网172.16.1.0/24的数据包将会被语句20和30转发到错误的接口。图14-4说明了为什么有必要配置第二行。

Schroeder的OSPF Hello数据包长度为44字节，如果没有包括语句10，那么OSPF Hello数据包将匹配到语句30并被转发到Pigpen，这将切断Lucy与Schroeder之间的邻接关系。如果匹配到语句10，OSPF数据包将被允许且按正常路由转发，邻接关系不会发生改变。

img654_1




图14-4 可以在分析仪中看到**OSPF Hello**数据包的长度

14.2.2 案例研究：策略路由选择和服务质量路由选择

虽然服务质量（QoS）路由选择超出了本卷的范围，但是这里必须注意，策略路由选择可以是QoS的一个重要组成部分。带有QoS的策略路由选择可以在数据包进入路由器接口时，通过设置数据包IP头内字段中的优先级和服务类型位来实现。图14-5给出了ToS字段的位信息。虽然在现代网络中很少使用ToS位，但是在QoS应用中优先级位焕发出新的生命力。ToS位可用于改变路由器为数据包选择的路由。而优先级位则用于区分路由器中数据包的优先次序。

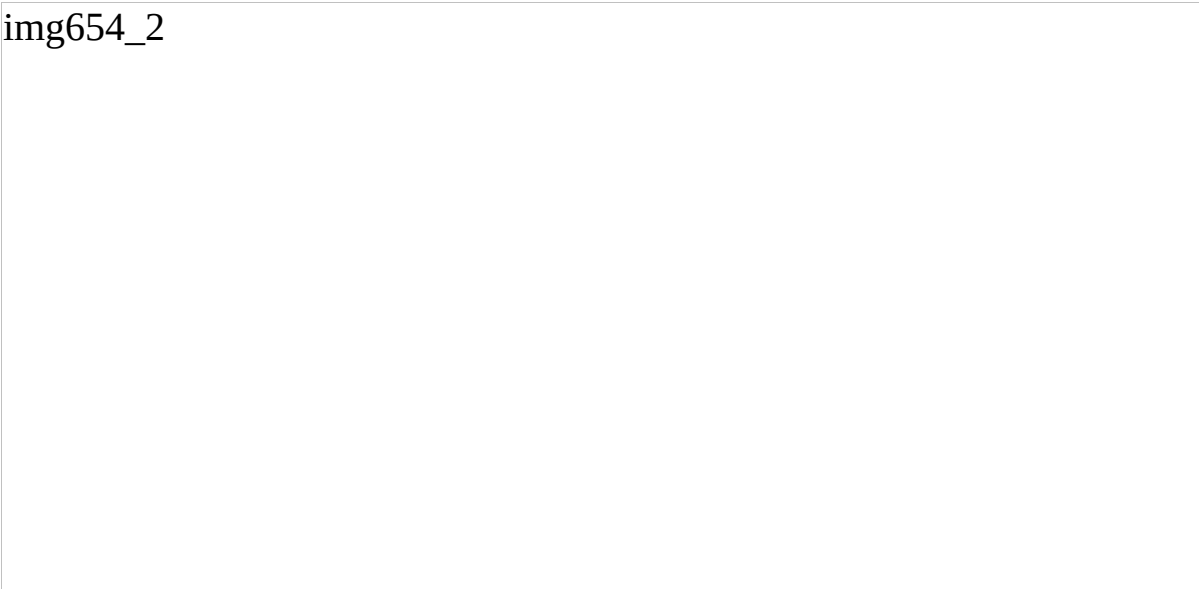


图14-5 IP报头服务类型字段中的优先级和ToS位

在路由映射中使用**set ip precedence** 语句可以设置优先级位。通过指定3个优先级位的十进制等价数或关键字可以对优先级进行设置。表14-5给出了会用到的十进制数和关键字。

表14-5 命令**set ip precedence**使用的优先级值和关键字

比特	数字	关键字
000	0	路由
001	1	优先级
010	2	立即
011	3	火速
100	4	火速—覆盖
101	5	紧急
110	6	Internet
111	7	网络

ToS位可以通过语句**set ip tos** 来设置；同优先级语句一样，语句的参数可以是数字和关键字，如表14-6所示。与优先级不同的是，你可以使用组合ToS值。例如，指定ToS为12（1100b）表明最小带宽和最大吞吐量。由于仅能使用一个关键字，因此为了设置组合ToS值，必须使用数

字。

表14-6 命令set ip tos用到的ToS值和关键字

比特	数字（0~15）	关键字
0000	0	正常
0001	1	最小开销
0010	2	最大可靠性
0100	4	最大吞吐量
1000	8	最小时延

图14-6给出了在QoS路由选择中怎样使用策略路由的例子。这里，路由器Pogo在网络OkefenokeeNet的边界。通过在Pogo的串行链路上配置策略路由，可以改变入站数据包的优先级位或ToS位，把IP流量区别为几种流量类型。具体配置见示例14-19。



图14-6 策略路由可以对进入网络数据包的优先级位和ToS位进行设置，网络中的路由器将基于对这些位的设置进行QoS决策

示例14-19 在Pogo设置优先级和ToS位

img656_1

语句10说明如果数据包匹配到访问列表1和110，那么优先级被设置为紧急。注意语句20没有**match** 语句，那么该语句将匹配所有在语句10没有匹配成功的数据包。在语句20中还有两个**set** 语句，这些语句将设置ToS位为最小延迟和最大可靠性，并且设置优先级为优先。图14-7给出了在OkefenokeeNet中捕获到的数据包，该数据包已经被Pogo上的路由映射修改过了。

在设置好进入网络数据包的优先级和ToS位之后，网络内的路由器将基于这些位所定义的部分或全部服务类别进行QoS决策。例如，为了对流量划分优先等级，可以根据优先级和ToS位配置优先级、自定义或加权公平队列。在某些实现中，优先级可被用于拥塞控制机制，例如加权随

机早期检测（Weighted Random Early Detection, WRED）。或者通过配置访问列表，使其可以基于优先级或ToS位允许和拒绝数据包经过某条链路，以便实现粗服务类别路由选择（Class of Service, CoS）。




图14-7 Pogo的策略路由将这个数据包的优先级位设置为**001b**, ToS位设置为最小延迟和最大可靠性（**1010b**）

14.2.3 案例研究：路由映射和重新分配

在IPv4和IPv6协议下，通过在**redistribute**命令中添加对路由映射的调用就可以使路由映射和重新分配一起使用。在图14-8给出的网络中，在路由器Zippy上，IS-IS的IPv4路由和OSPF路由正在进行相互的路由重新分配。在图示中列出的网络和子网的地址中，仅第3个八位组字节为奇数的子网被重新分配。

img657_1



图**14-8** **OSPF**和**IS-IS**正在进行相互的路由重新分配。**redistribute**命令和路由映射一起被用作简单的路由过滤，或者说它们可以用于修改被重新分配路由的属性

Zippy的配置见示例**14-20**。

示例**14-20** **Zippy**仅对第**3**个八位组字节为奇数的地址进行重新分配

img657_2

路由映射Griffy和Toad执行相同的功能，但是它们的逻辑却不相同。Griffy使用否定逻辑，它标识那些不被重新分配的路由，而Toad使用肯

定逻辑，它标识将要被重新分配的路由。

Griffy的语句10拒绝访问列表1许可的任何路由（第3个八位组字节为偶数的地址）。因为语句10不能匹配到第3个八位组字节为奇数的地址，因此它们被传递到语句20。语句20没有**match** 命令，因而缺省匹配所有地址。语句20具有允许操作，所以许可奇数地址的路由。结果如示例14-21所示。

示例14-21 Shelflife路由表包含的**IS-IS**域内的目标网络，第**3**个八位组字节都为奇数

img658_1

路由映射有一条单一语句允许访问列表2许可的路由（第3个八位组字节为奇数）。第3个八位组字节为偶数的地址在访问列表2中寻找不到匹配。当重新分配时，缺省的路由映射语句是拒绝所有路由，所以不能被访问列表2匹配的地址将不会被重新分配。示例14-22给出了路由映射Toad的结果。

示例**14-22** **Zerbina**路由表中包含的**OSPF**域内的目标网络，第**3**个八位组字节为奇数

img658_2

另一个配置将实现相同的目的，例如路由映射Toad使用示例14-23的访问列表将起到同样的作用。

示例**14-23** **Zippy**上路由映射**Toad**的另一种配置

img659_1

虽然路由映射可以像简单的路由过滤一样工作得很好，但是它的能力体现在可以按照多种方式改变路由。考虑示例14-24中Zippy的配置。

示例**14-24** **Zippy**的路由映射配置设置了度量类型、度量值和被重新分配路由的层级

img659_2

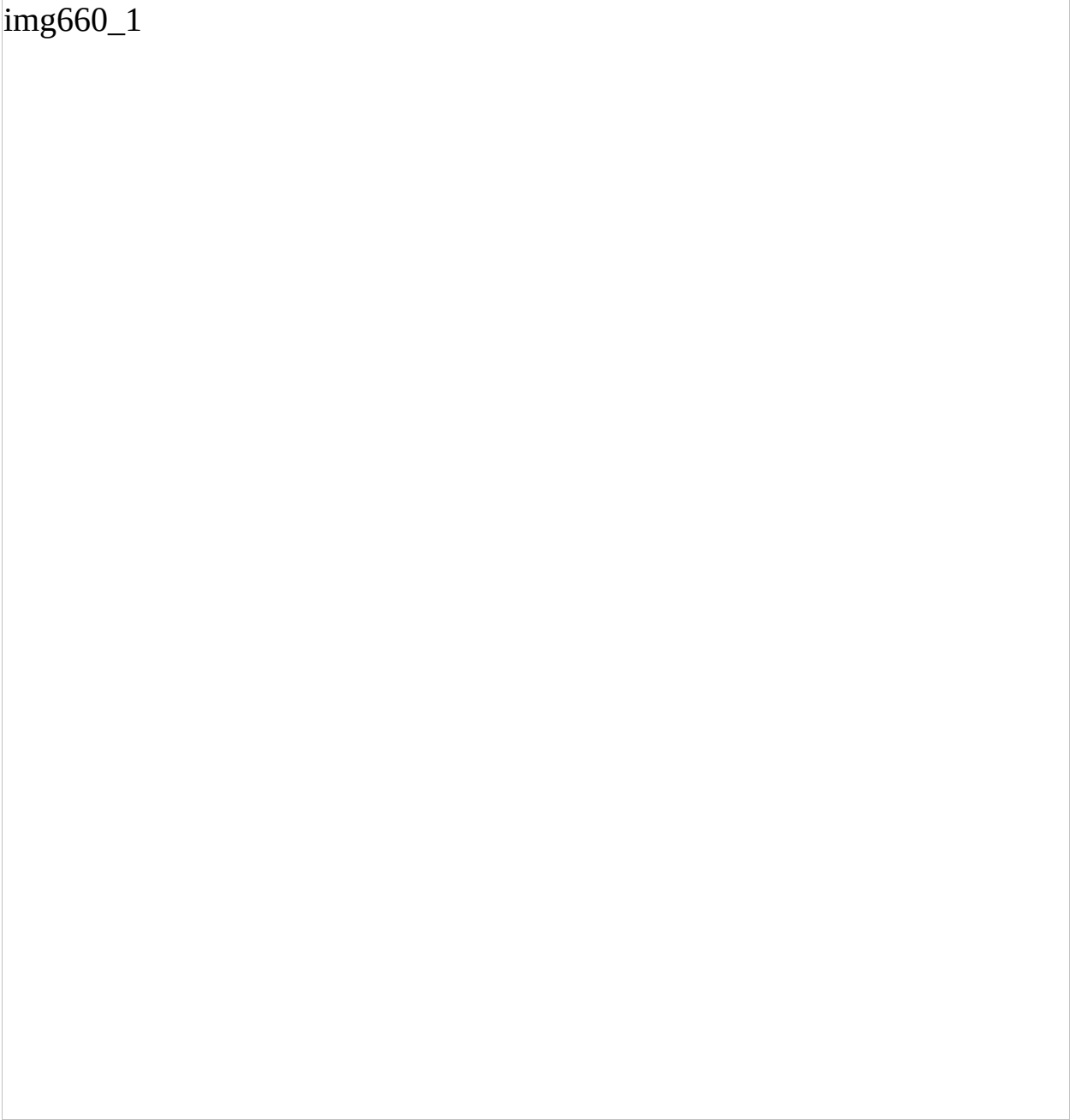
路由映射Griffy中语句10允许访问列表1中地址的路由，并且将它们作为外部类型1被重新分配到OSPF。语句20允许所有其他路由，并把它们作为外部类型2重新分配。结果见示例14-25。

路由映射Toad中语句10允许指向访问列表2所定义的地址的路由，并且这些路由作为L1路由被重新分配到IS-IS，度量值为15。语句20允许所有其他路由，在IS-IS配置下的**redistribute** 命令把这些路由作为L2路由重新分配，度量值为25（参见示例14-26）。

示例**14-25** 如果路由的目标网络在**IS-IS**域，且地址的第**3**个八位组字节为偶数，则路由为**E1**，如果为奇数，则为**E2**

img659_3

img660_1



示例14-26 如果路由的目标网络在**OSPF**域且地址的第**3**个八位组字节为奇数，则路由为**L1**，路由度量为**15**；如果为偶数，则为**L2**。路由度量为**25**（加**10**是因为从**Zippy**到**Zebbina**的跳数为**10**）

img660_2

14.2.4 案例研究：路由标记

图14-9表明，来自多个路由选择域的路由被重新分配到一个运行OSPF的传输域，其中每个域都运行单独的路由选择协议。在OSPF域的另一边，路由必须被重新分配回到它们各自的域。在从OSPF域进入每个域的出口点，可以使用路由过滤器以允许仅属于该域的路由通过。然而，如果每个域的路由很多或变动很大，那么路由过滤器也可能会很难管理。

img661_1




图14-9 图左边3个域的路由被重新分配到一个运行**OSPF**的传输域。图右边域的路由必须被重新分配回产生它们的域

处理这个问题的另一种方法是，在OSPF传输域的入口对路由进行标记（tag），该标记在每个域内均是惟一的。在出口处，我们可以借助标记重新分配路由，而不通过明确的地址。传输网络的路由选择协议没有必要使用该标记，而仅仅是向外部网络来回传送它们。RIPv2、EIGRP、集成IS-IS和OSPF都支持路由标记。BGP也支持路由标记。RIPv1不支持标记。本节中有一个案例研究会讨论运行OSPF的传输网络如何使用路由标记。

回顾一下第6章、第7章、第8章和第10章的数据包格式，可以看出RIPv2

消息支持16位标记，IS-IS域间路由协议信息TLV也支持16为标记，而EIGRP外部路由TLV和OSPF 5型LSA支持32位标记。这些标记可以用十进制数表示，因此RIPv2所携带的标记值在0～65535之间，EIGRP和OSPF携带的标记值在0～4 294 967 295之间。

在图14-10中，路由器Dagwood正在接受来自3个不同路由域的路由，并且把它们重新分配到OSPF域。目标是标记来自每个域的路由以便在OSPF域内可以标识它们的源点域。来自域1的路由标记为1，域2的标记为2，等等。



图14-10 配置Dagwood，使得来自这3个域的路由在被重新分配到OSPF时被标记


Dagwood配置如示例14-27所示。

示例14-27 当路由从RIP和EIGRP被重新分配到OSPF时，Dagwood对这些路由进行标记

img662_1

首先，注意OSPF配置下的命令**redistribute igrp**。Dagwood接受的EIGRP路由仅来自一个EIGRP域，所以直接在**redistribute**命令上设置标记为1。然而，RIP路由是来自两个RIP域的，因此这里需要路由映射。路由映射Dithers设置RIP路由的标记为2或3，具体依赖于路由是学自Funky（10.1.2.3）还是Beetle（10.1.2.4）。图14-11给出了一个LSA通告，被通告的路由是从RIP那里学到的路由之一，标记为2。

img662_2



图**14-11** 这个类型**5**的**LSA**正在通告域**2**内的网络**192.168.2.0**，域**2**在**OSPF**域内。路由标记见最后一行

在**OSPF**的链路状态数据库中也可以观察到路由标记（参见示例**14-28**）。

示例**14-28** **OSPF**链路状态数据库指明了每个外部路由的标记，该标记是由**Dagwood**的重新分配进程设置的

img662_3



img663_1

在图14-12中，Blondie仅可以向Alley重新分配域2的路由，向Oop重新分配域1的路由。因为这些路由在进入OSPF传输域时已经被标记过，按示例14-29的方式很容易实现这些功能。

img663_2

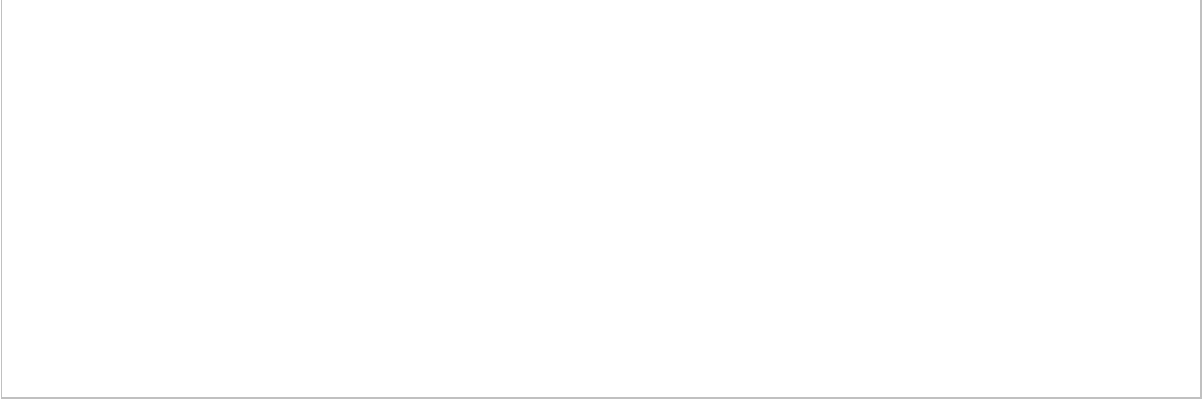


图14-12 **Blondie**使用路由映射并根据路由标记重新分配路由

示例14-29 **Blondie**使用路由映射向**EIGRP**重新分配标记为**1**的路由，
向**RIP**重新分配标记为**2**的路由

img663_3

示例14-30给出了在Alley和Oop上的路由。使用路由标记过滤路由的一个缺点是不能通过接口过滤路由。例如，如果Blondie必须向域2和域3发送路由，而且这两个域都运行RIP。则不可能通过配置使路由映射向一个RIP进程发送某些路由，而向另一个RIP进程发送其他的路由。这些路由必须使用命令**distribute-list** 借助地址来过滤。

示例**14-30** 在图**14-12**中，**Alley**和**Oop**的路由表显示了**Blondie**的配置结果

img664_1

14.2.5 案例研究：从OSPF路由表中滤掉被标记的路由

图14-10和图14-12中运行OSPF协议的网路都是一个传输网路。如果传输区中设备不需要向任何其他域发送数据包，那么在OSPF的路由表中就不必维护这些域的网络。因此OSPF路由器可以使用标记路由、分布列表和路由映射来阻止这些网络被添加到路由表中，而又不会影响到链路状态数据库中的表项。

按照示例14-31对OSPF域内的路由器Sally进行修改。

示例**14-31** Sally使用标记把路由从OSPF路由表中过滤掉

img665_1

路由映射Charlie禁止标记为1、2和3的地址网路，因此命令**distribute-list in** 会把这些地址从路由表中删去。被路由映射语句20允许的所有其他地址将被添加到路由表中。在执行重新分配的边界路由器Blondie和Dagwood上没有应用这个分布列表。如果一个地址不在路由表中，即使它存在于OSPF的LSA数据库中，路由器也不会把它重新分配到其他路由选择协议。示例14-32给出了Sally的路由表和OSPF的LSA数据库。

示例**14-32** 带有标记的OSPF地址被从路由表中过滤掉，然而这些地址仍存在于OSPF的LSA的数据库中

img665_2

Sally的IP路由表中不再包含标记为1、2和3的路由。

14.2.6 案例研究：使用路由映射重新分配IPv6路由

IPv6路由选择协议同样支持使用路由映射重新分配路由。配置方法与IPv4基本相同。

如图14-13所示，在图14-10中的网络上添加了IPv6地址和路由选择协议。Funky和Beetle都运行了RIPng。Dagwood在RIPng和IS-IS之间重新分配IPv6前缀。仅仅是来自Beetel的IPv6前缀被重新分配到IS-IS中，其中前缀2001:db8:0:77::/64的度量值为10，2001:db8:0:200::/64的度量值为100。

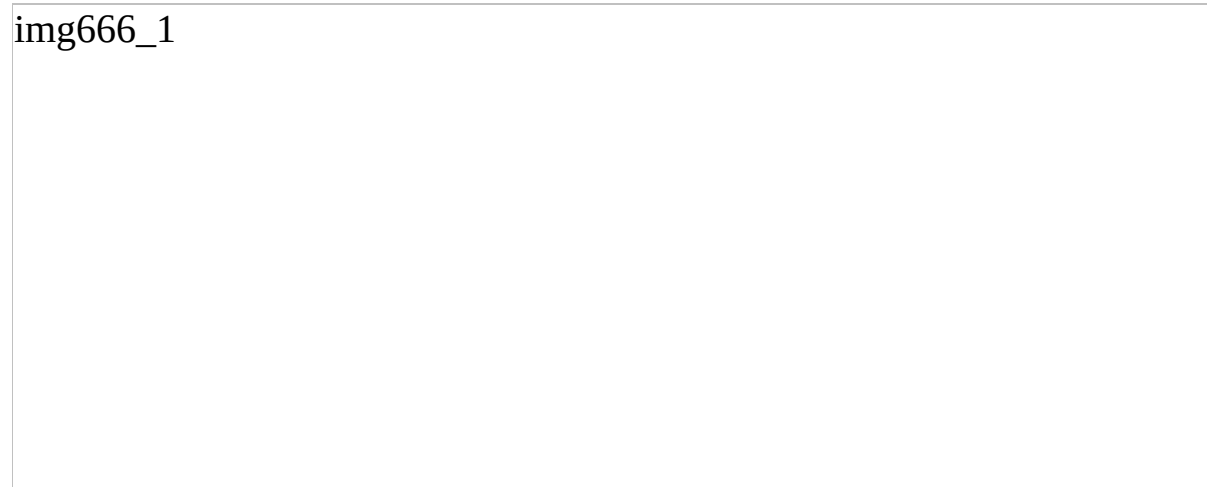


图14-13 在图14-10的网络中添加了IPv6，并且IPv6前缀在RIPng和IS-IS之间被重新分配

Dagwood的配置见示例14-33。

示例14-33 Dagwood的IPv6配置

img666_2

我们可以使用IPv6的前缀列表来匹配路由信息的源点或为重新分配指定的地址。**Beetlefilter**的语句10指明路由的源地址必须是**Beetle**的IPv6地址，而且前缀地址必须是2001:db8:0:77::/64，一旦匹配成功，度量值将被设置为10。如果匹配失败，执行语句20。如果源地址是**Beetel**，前缀为2001:db8:0:200::/64，那么度量值设置为100。如果源地址不是**Beetel**或前缀也不匹配，那么该路由将不会被重新分配。

Sally的IS-IS（参见示例14-34）数据库给出了被重新分配的前缀。

示例14-34 可以从**Sally**的IS-IS数据库中看到被重新分配的路由

img667_1

14.3 展望

本章结束了本书对TCP/IP路由技术中内部网关协议的深入讨论，如果你正准备成为CCIE，那么你当然想在考试之前知道本书的主题。你可以使用每章后面的复习题和配置练习测试一下你的理解能力和准备水平。如果你没有学习过有关外部网关协议的TCP/IP路由技术，那么在你下一步安排中，学习这部分内容是较为合理的。

14.4 总结表：第14章命令总结

img667_2



img668_1



14.5 复习题

1. 路由映射有哪些方面类似于访问列表？两者有什么不同？
2. 策略路由是什么？
3. 路由标记是什么？
4. 路由标记以什么样的方式影响路由选择协议？

14.6 配置练习

1. 在图14-14中，请为路由器A配置策略路由，使得从子网172.168.1.0/28出发经过172.16.1.112/28到达路由器A的数据包被转发到路由器D，而从子网172.16.1.128/28出发经172.16.1.240/28到达路由器A的报文被转发到路由器E。
2. 在图14-14中，为路由器A配置策略路由，使得从子网172.16.1.64/28出发经过172.16.1.112/28到达路由器C的数据包被转发到路由器D；而相同的数据包如果到达路由器B，则被转发到路由器E。所有其他数据包将被正常地转发。
3. 在图14-14中，为路由器A配置策略路由，使得所有经172.16.1.240/28去往子网172.16.1.0/28且源端口为SMTP的数据包被转发到路由器C，任何其他去往相同子网的UDP数据包被转发到路由器B。通过策略路由或常规路由选择协议没有向路由器C或路由器B转发其他的数据包。

img669_1



图14-14 配置练习1~3中的网络

4. 在图14-15中，路由器的OSPF和EIGRP的配置如下：

```
router eigrp 1
```

```
network 192.168.100.0
```

```
!
```

```
router ospf 1
```

```
network 192.168.1.0 0.0.0.255 area 16
```

配置路由器把内部EIGRP路由作为度量为10的E1路由向OSPF重新分配，把外部EIGRP路由作为度量为50的E2路由向OSPF重新分配。EIGRP域内的所有网络和子网除了10.201.100.0/24外，都将被重新分配。

5. 配置图14-15中的路由器，向EIGRP重新分配内部OSPF路由，其中内部OSPF路由的时延要小于外部OSPF路由的时延。仅允许OSPF域内的3个C类网络被重新分配。

img670_1




图14-15 练习4和练习5的路由器配置

14.7 故障诊断练习

已知下面配置：

```
interface TokenRing1
ip address 192.168.15.254 255.255.255.0
ip policy route-map Ex1
!
access-list 1 permit 192.168.0.0 0.0.255.255
access-list 101 permit host 192.168.10.5 any eq telnet
!
route-map Ex1 permit 5
match ip address 1
set ip next-hop 192.168.16.254
!
route-map Ex1 permit 10
match ip address 101
set ip next-hop 192.168.17.254
```

上面配置的目的是对所有源地址前缀在192.168.0.0～192.168.255.255中的数据包进行策略路由。除了来自主机192.168.10.5的Telnet数据包外，所有其他数据包将被转发到192.168.17.254。在配置中有两个错误导致策略路由工作不正常，请问错误是什么？

[1] 还要注意一点，在配置中没有指定操作，所以缺省操作“许可”出现在最终的配置中。

[2] 在一些IOS版本中，如果你在修改路由映射的匹配和设置语句时没有指定序列号，那么将会产生一条IOS消息——“% Please specify the entry by its sequence”，而不是假定去修改与序号10相关的语句。

[3] 注意命令 **debug ip packet** 参考了访问列表5。为了使调试功能对不感兴趣的流量不作显示，该访问列表仅允许连接在路由器Charlie的子网。

第四部分

附录

附录A 教程：二进制和十六进制

附录B 教程：访问列表

附录C CCIE备考提示

附录D 复习题答案

附录E 配置练习答案

附录F 故障诊断练习答案

附录A

教程：二进制和十六进制

理解二进制和十六进制的最佳方法是先透彻领悟十进制计数系统。十进制（decimal）系统是基于10的计数系统（词根deci-表示10）。“基于10”指的是由10个数位（digit）0到9来表示数。通常我们使用十进制，这是因为我们的祖先用他们的手指（finger）来计算牛、孩子、敌人（事实上digit的意思就是finger）。

使用“位值（place value）”，可以用不多的几个数位（如10个十进制数位）来表示很大的数。所有计数系统的位值从最右边开始，是基数的0次幂。从右往左，基数的幂依次增大1：

$$B^4 \ B^3 \ B^2 \ B^1 \ B^0$$

基数是10时，前5个位值是：

$$10^4 \ 10^3 \ 10^2 \ 10^1 \ 10^0$$

对任何基数，前两个位值是最容易计算的。任何数的0次幂是1，所以 $10^0=1$ 。任何数的1次幂就是它本身，所以 $10^1=10$ 。第三个位值也是容易计算的，只要简单地用第二个位值乘以基数就可以。事实上，每一个位值都可以用它前边一个位值乘以基数计算出来。所以上面5个位值是：

$$10^0=1$$

$$10^1=1 \times 10=10$$

$$10^2=10 \times 10=100$$

$$10^3=100 \times 10=1000$$

$$10^4 = 1000 \times 10 = 10000$$

所以，对于基数是10的计数系统，前5个位值是：

10000 1000 100 10 1

根据位值来读一个数，比如57258，指的是有5个10000，7个1000，2个100，5个10，以及8个1。就是说：

$$5 \times 10\,000 = 50\,000$$

$$7 \times 1000 = 7000$$

$$2 \times 100 = 200$$

$$5 \times 10 = 50$$

$$8 \times 1 = 8$$

将这些结果相加，结果是 $50000 + 7000 + 200 + 50 + 8 = 57258$ 。

我们对十进制都非常熟悉，所以，我们很少会去考虑将一个数分解成位值。但是，这种方法对于阐明其他进制的数是非常至关重要的。

A.1 二进制数

计算机从最底层来看，只不过是电子开关的集合而已。而数字和字符是由这些开关的状态来表示的。由于一个开关仅有两种状态——开或者关，所以它使用二进制（binary），或者说基数为2的计数系统（词根*bi*表示2）。一个基数为2的系统仅仅有两个数位：0和1。计算机通常将这两个数位集成8个位值，即一个字节（byte）或八位组字节（octet）。这8个位值是：

$$2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$$

位值这样计算：

$$2^0 = 1$$

$$2^1 = 1 \times 2 = 2$$

$$2^2 = 2 \times 2 = 4$$

$$2^3 = 4 \times 2 = 8$$

$$2^4 = 8 \times 2 = 16$$

$$2^5 = 16 \times 2 = 32$$

$$2^6 = 32 \times 2 = 64$$

$$2^7 = 64 \times 2 = 128$$

所以一个二进制八位组的位值是： 128 64 32 16 8 4 2 1

因此，二进制八位组10010111可以这样理解：

$$1 \times 128 = 128$$

$$0 \times 64 = 0$$

$$0 \times 32 = 0$$

$$1 \times 16 = 16$$

$$0 \times 8 = 0$$

$$1 \times 4 = 4$$

$$1 \times 2 = 2$$

$$1 \times 1 = 1$$

$$\text{或者 } 128 + 16 + 4 + 2 + 1 = 151$$

对于二进制数，因为每一个位值要么就是该值本身，要么就没有，所以比较简单。另外一个例子： $11101001 = 128 + 64 + 32 + 8 + 1 = 233$ 。就是说，将二进制转为十进制仅仅是一个将位值相加的过程，将十进制转为二进制仅仅是将位值相减的过程。例如，要将十进制数178转为二进制，首先把178减去最高的位值：

1. 178大于128，所以我们就知道在该位值上有一个1： $178 - 128 = 50$ 。
2. 50比64小，该位值上有一个0。
3. 50比32大，所以该位值上有一个1： $50 - 32 = 18$ 。
4. 18比16大，该位值上有一个1： $18 - 16 = 2$ 。
5. 2比8小，该位值上有一个0。
6. 2比4小，该位值上有一个0。
7. 2等于2，该位值上有一个1： $2 - 2 = 0$ 。
8. 0小于1，该位值上有一个0。

把这些步骤的结果综合起来，用二进制表示178就是10110010。

另外一个例子可能会有帮助。给出110：

1. 110比128小，所以在位值上有一个0。
2. 110比64大，所以在位值上有一个1： $110-64=46$ 。
3. 46比32大，所以在位值上有一个1： $46-32=14$ 。
4. 14比16小，所以在位值上有一个0。
5. 14比8大，所以在位值上有一个1： $14-8=6$ 。
6. 6比4大，所以在位值上有一个1： $6-4=2$ 。
7. 第2个位值上有一个1： $2-2=0$ 。
8. 0比1小，所以在位值上有一个0。

所以，110用二进制表示就是01101110。

A.2 十六进制数

写一个二进制八位组并不有趣。对于经常要使用这些数字的人来说，受欢迎的是更简洁的表示法。一个可能的表示法是为每一个可能的八位组分配一个单独的字符。但是，8位有 $2^8=256$ 种不同的组合，所以，用单独的字符表示所有八位组需要256个数位，或者说一个基数为256的计数系统。

将一个八位组看作是两个各4位的组合或许会更简单一些。例如，11010011可以看作是1101和0011。对4个位来说，有 $2^4=16$ 种不同的组合，所以有基数16，或者说十六进制（hexadecimal）计数系统，一个八位组可以用两位来表示（词根*hex* 的意思是“six”，*deci* 的意思是“ten”）。表A.1列出了十六进制数以及相应的十进制数和二进制数。

表A-1 十六、十和二进制数

十六进制	十进制	二进制
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

因为十六进制和十进制的前10个数字是一样的，所以我们有意在一个十六进制数前面加0x，或者在后面加一个h，以便和十进制数区分开。例如，十六进制数25应该写成0x25或者25h。本书使用0x表示法。

刚才学过二进制的表示法，很容易写出一个4位二进制数的十进制表达形式。同样也很容易将一个十进制数转为十六进制。于是，我们可以很容易地通过3个步骤将一个二进制八位组转为十六进制：

1. 将八位组分成2个4位的二进制数。
2. 将每个4位二进制数转为十进制。
3. 把每个十进制数用十六进制来表示。

例如：把11010011转为十六进制：

1. 11010011变成1101和0011。
2. $1101=8+4+1=13$ ， $0011=2+1=3$ 。
3. $13=0xD$ ， $3=0x3$ 。

所以，11010011用十六进制表示就是0xD3。

把十六进制转为二进制是上述3步的简单逆序。例如，把0x7B转为二进制：

1. $0x7=7$ ， $0xB=11$ 。
2. $7=0111$ ， $11=1011$ 。
3. 把2个4位二进制数写在一起就是 $0x7B=01111011$ ，十进制为123。

附录B

教程：访问列表

目前对访问列表的命名可能并不是很恰当，访问列表最初的目的是许可或拒绝访问进入、离开或经过路由器的数据包。而现在访问列表已成为控制帧和数据包行为的强大工具，它们的用途可分为3类（如图B-1所示）。

img681_1



图B-1 访问列表可以用做安全过滤器、流量过滤器或用于数据包标识

- 安全过滤器可以保护路由器及路由器传递流量所到达网络的完整性。一台安全过滤器许可少数清晰的数据包通过，而拒绝其他任何数据包通过。
- 流量过滤器可以阻止不必要的数据包通过带宽有限的链路。这些过滤器的外形和行为同安全过滤器很相似，但是逻辑一般是颠倒的：流量过滤器拒绝少数不必要的数据包而许可其他任何数据包通过。
- 在Cisco路由器上的许多工具要求数据包验证，例如拨号列表、路由过滤器、路由映射和队列列表，都必须能够标识确定的数据包。访问列表可以链接到这些或其他工具上，并且提供数据包标识功能。

B.1 访问列表基础知识

一个访问列表是一组按顺序排列的过滤器。每台过滤器是由匹配标准和一个过滤动作构成的。过滤动作不是许可就是拒绝；而过滤标准既可以用像源地址一样简单的参数，也可以复杂到使用诸如源和目的地址、协议类型、端口号或套接字（**Socket**）和某些标记状态（如**TCP ACK**位）等参数的组合。

一个数据包从栈顶进入过滤器（如图B-2所示），在每台过滤器中匹配的标准将被应用，如果发生匹配，那么指定的许可或拒绝动作将被执行。如果匹配没有发生，数据包将向下移动到栈中下一台过滤器并再次重复匹配过程。

img682_1



图B-2 一个访问列表是一组按顺序排列的过滤器，每台过滤器定义了一种匹配标准和一个过滤动作

在图B-2中，许可意味着数据包将被允许离开接口E0；拒绝意味着数据包将被丢弃。例如，源地址为主机A的数据包将在第一台过滤器中被丢弃。假设数据包的源地址是网络5中子网2上的主机D，而第一个过滤器指明了对主机A的匹配标准，因此没有匹配发生，数据包进入第二层。在第二台过滤器中又指明了子网3，因此还是没有匹配发生。数据包进入第三台过滤器，该过滤器指明了网络5，因此匹配成功。由于本层的动作是许可，所以数据包被允许离开接口E0。

B.1.1 隐式拒绝一切

如果数据包经过所有过滤器都没有发生匹配，那会出现什么情况？在这种情况下路由器必须知道该如何处理数据包，也就是必须有一个缺省动作。缺省动作可以是允许所有没有匹配的数据包通过或者是拒绝它们通过。Cisco选择了拒绝它们通过：对任何经过访问列表的数据包，如果没有发生匹配将会被自动丢弃。

这种方法是一个正确的工程选择，特别是当访问列表用于安全控制时。丢弃那些本不应该被丢弃的数据包，比起许可那些你因疏忽而没有进行过滤的数据包应该更好。

最后一台过滤器被称为隐式地拒绝一切（implicit deny any）（如图B-3所示）。正如其名字所暗示的，在你创建的任何一个访问列表中都不会显示该过滤器。它仅仅是一种缺省动作，并且它在所有访问列表中都处于最后。

img683_1

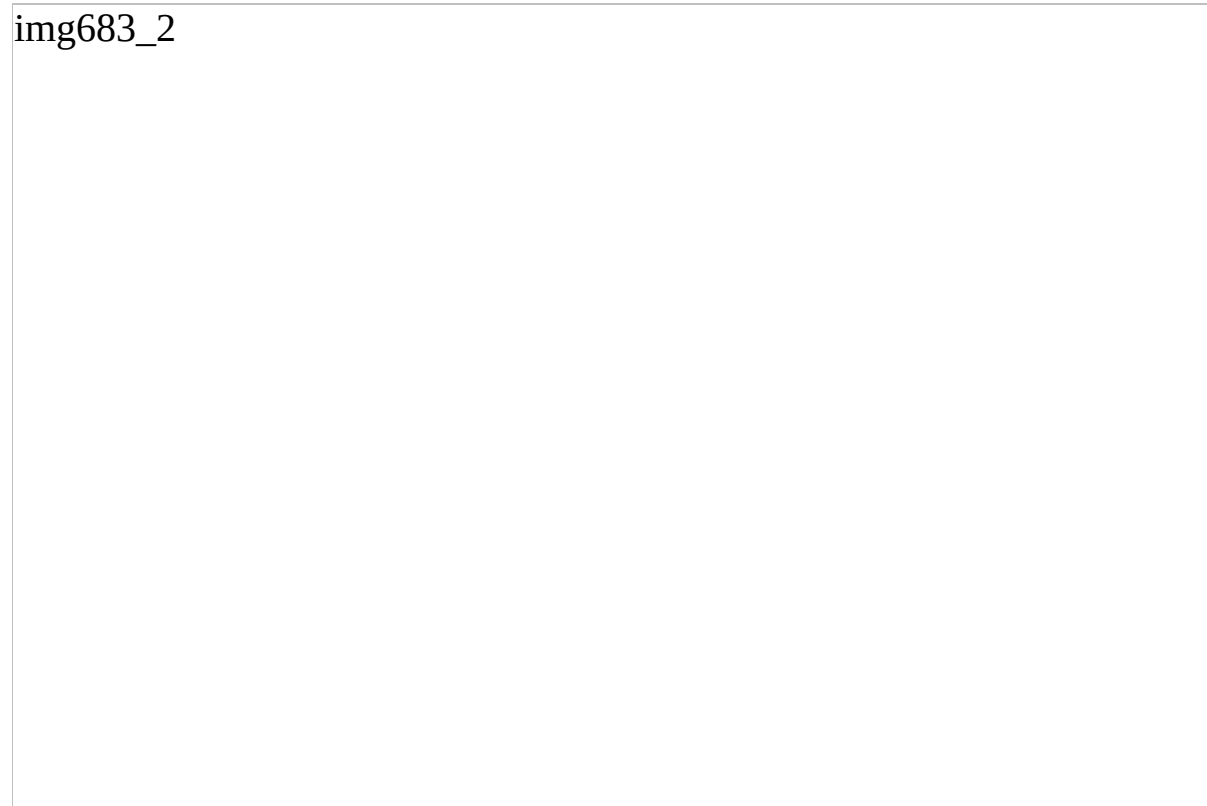
图B-3 所有以隐式拒绝一切结尾的访问列表将丢弃那些在表中没有匹配的数据包

通过在最后一行建立显式地许可一切（**explicit permit any**）可以覆盖这个缺省动作。这里隐含了一点，即经过其他所有过滤器的数据包在到达缺省的拒绝一切动作之前，会匹配到许可一切动作，因而没有匹配到任何动作的数据包将被许可—不会有数据包到达隐式地拒绝动作。

B.1.2 顺序性

访问列表是从上到下按顺序执行的。这一概念很重要：或许访问列表发生故障的最普遍原因就是，过滤器的放置顺序出现错误。按顺序的访问列表的第一个匹配项总是被执行的。在第一个匹配项匹配成功后，访问列表其余的匹配项将被忽略。

在图B-4中，子网10.23.147.0/24应该被拒绝，而网络10.0.0.0其余的子网应该被许可。左边的访问列表顺序有错误，网络10.0.0.0（包括它的子网10.23.147.0）将匹配到第1行并且被许可。而被拒绝的子网数据包根本不会到达第2行。



图B-4 如果访问列表中的个别过滤器没有按照正确的顺序配置，那么访问列表将不能正常工作

右边的访问列表是准确的。子网10.23.147.0匹配到第1行并且被拒绝，而10.0.0.0的其余子网将会匹配到第2行并且被许可。

B.1.3 访问列表类型

图B-4中右边的访问列表的实际配置参见示例B-1

示例B-1 图B-4的第二个访问列表显示，示例中的每一行都代表了每过滤器层

img684_1

一个配置行表示访问列表的每过滤器层。这里将简略地讨论一下访问列表中的各种组件，但请先注意在两行中都出现的数字9表示是访问列表编号，它主要有两个用途：

- 将访问列表中的所有行都连接在一些，使得该访问列表区别于路由器配置中的其他访问列表（通常在一个路由器中存在多个访问列表）。
- 路由器必须有区分访问列表类型的方法。Cisco IOS软件有IP、IPX、AppleTalk、DEC、NetBIOS、桥接和许多其他协议的访问列表，而且其中许多协议都有多种访问列表类型。访问列表编号可以表明路由器访问列表的类型。

访问列表类型可以通过数字和名字来标识。表B-1给出了被编号的访问列表类型和每种类型的访问列表可用的编号范围。例如，因为编号1010在1000～1099之间，所以**access list 1010**标识了IPX SAP。

表B-1 Cisco访问列表编号

访问列表类型	范围
标准IP	1～99， 1300～1999
扩展IP	100～199， 2000～2699
以太网类型代码	200～299
以太网地址	700～799
透明桥接（协议类型）	200～299
透明桥接（厂商代码）	700～799
扩展的透明桥接	1100～1199

DECnet和扩展的DECnet	300～399
XNS	400～499
扩展XNS	500～599
AppleTalk	600～699
源路由桥接（协议类型）	200～299
源路由桥接（厂商代码）	700～799
标准IPX	800～899
扩展IPX	900～999
IPX SAP	1000～1099
NLSP路由汇总	1200～1299
标准VINES	1～99
扩展VINES	100～199
简单VINES	200～299

在一个范围内，访问列表编号不必按照任何特殊顺序，也就是说，在路由器中不必按第一个标准IP列表为1，第二个为2等进行编号。它们可以使用1～99，或者1300～1399中任意一个数字，只要保证在一台路由器中每个访问列表编号惟一就可以。

此外，请注意，不同协议的一些编号范围是相同的，例如以太网类型代码、源路由桥接和简单VINES。在这种情况下，路由器将会通过访问列表自身的格式来区分访问列表类型。可以使用名字代替数字来标识下面的访问列表类型：

- Apollo域；
- 标准IP；
- 扩展IP；
- ISO CLNS；
- 源路由桥接NetBIOS；
- 标准IPX；

- 扩展IPX;
- IPX Sap;
- IPX NetBIOS;
- NLSP路由汇总。

在示例B-2中，名字为Boo的访问列表用来标识IPX NetBIOS。

示例**B-2** 名字为**Boo**的访问列表拒绝了各种**NetBIOS**设备

img685_1

注意，虽然标准和扩展IP访问列表通常使用编号，但是它们也可以使用名字。在IOS 11.2及更高的版本中支持这一协定。在某些环境中，可能会使用大量IP列表配置路由器。用名字代替数字，可以更加容易地标识单独的访问列表。

命名IP访问列表当前仅可以与数据包和路由过滤器一起使用。更详细的信息请参考Cisco配置指南。

B.1.4 编辑访问列表

任何一位曾经从控制台上编辑过多行访问列表的人，都会告诉你这是一种在挫折中经受锻炼的过程。在IOS 12.2（14）版本之前，没有办法能向列表的中间添加一行。所有新输入的行都被添加到列表尾部。如果你发现输入错误并且试图删去其中特定的一行，例如，

```
no access-list 101 permit tcp 10.2.5.4 0.0.0.255 192.168.3.0 0.0.0.255  
eq 25
```

那么访问列表101的所有行将与这一行一起被删除。

更方便的办法是将访问列表剪切和粘贴到你PC的记事本，或者上载配置到TFTP服务器上，然后在那里编辑访问列表。当编辑结束后，新的访问列表将被载入路由器。这里提醒一下，所有新输入的行都会被添加到访问列表的尾部。记住，始终将**no access-list #**加在被编辑访问列表的开始，其中#是你正在编辑的访问列表编号。示例B-3显示了一个例子。

示例B-3 在PC或服务器上开始创建一个访问列表之前，先添加一条命令**no access-list**，这样加载到路由器上的访问列表每一次都是重新创建的

img686_1

no access-list 5 将会在添加新访问列表之前，从配置文件中删除旧的访问列表5。如果你省略了这一步，那么新列表仅仅会被添加到旧列表的结尾。

利用命令**show access-list**可以显示当前配置的列表，参见示例B-4所示。

示例B-4 利用命令**show access-list**可以显示路由器上当前配置的访问列表

img686_2

这里请注意每一个访问列表条目的数字。它们是序列号。在IOS 12.2（14）S版本后，序列号是自动增加到访问列表条目中的。这个序列号允许你在某个列表的顶端或中间插入一个条目。如果你不指定序列号，第一个条目将指定为10，后面跟着的每一个序列号都将增加10。当一台路由器重启时，序列号会重置为10、20、30等。序列号也允许我们在某个访问列表中删除指定的条目。

参见示例B-5中的配置，在访问列表5中，允许子网172.16.20.0中所有的主机通过的条目替代允许子网172.16.12.0中所有的主机通过的条目。

示例B-5 访问列表的更新可以通过使用序列号替换或增加条目实现

img686_3

在增加一个序列号为20的新条目之前必须首先删除原有的序列号为20的条目，否则将会出现序列号冲突的错误。

B.2 标准IP访问列表

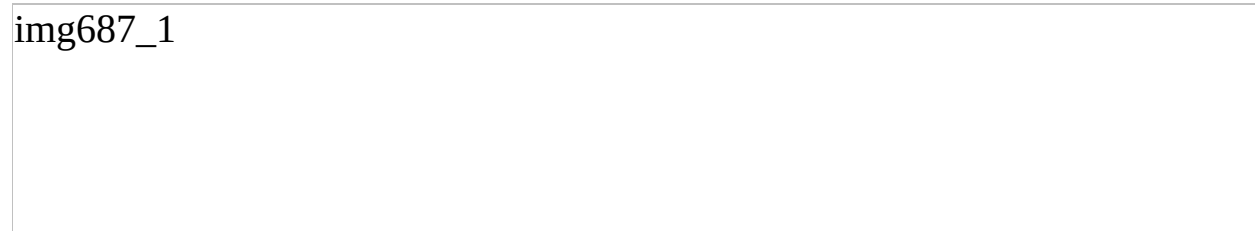
有两种进入访问列表的方式。其中之一是：

```
access-list access-list-number {deny|permit } source[source-wildcard]
```

另一个配置访问列表的方法是输入全局模式下的访问列表命令，也可以进入访问列表的配置模式。在访问列表的配置模式下，可以允许或拒绝数据包的通过，指定序列号和注释：

```
ip access-list standard {access-list-number | name }
```

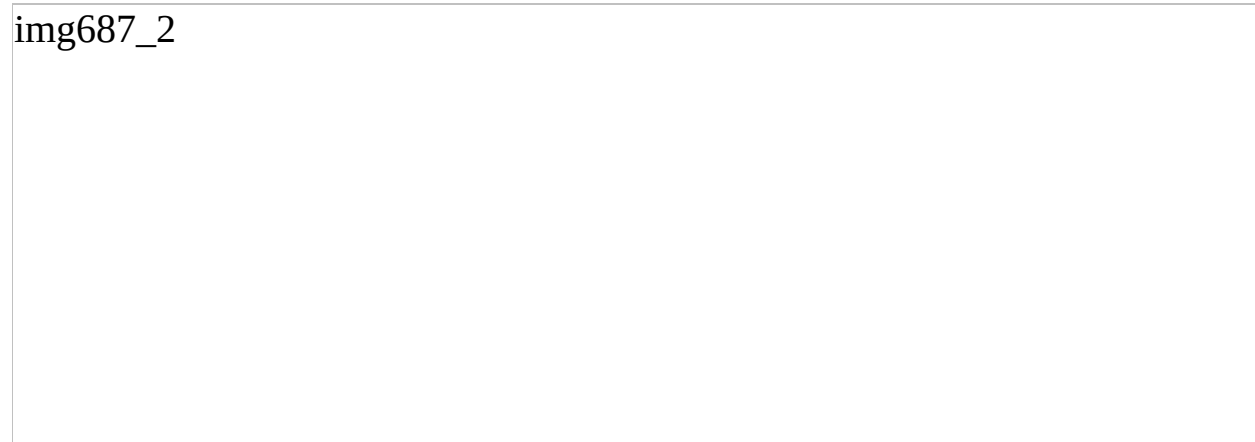
采用上面这条命令可以进入访问列表配置模式。有关标准IP访问列表的进一步配置选项是：



```
access-list 100 deny 10.10.10.0 0.0.0.255
```

这条命令根据表B-1在1～99和1300～1999之间指定了访问列表的编号、动作（许可和拒绝）、源IP地址、通配符（或反向）掩码。示例B-6中显示了一个标准IP访问列表的例子。

示例B-6 标准访问列表1允许和拒绝多个主机和子网地址



```
access-list 100 permit 10.10.10.0 0.0.0.255  
access-list 100 deny 10.10.10.0 0.0.0.255  
access-list 100 permit 10.10.10.0 0.0.0.255
```


例子中的前两行允许源地址属于指定主机172.22.30.6和172.22.30.96的数据包通过。这看似很恰当，尽管反码0.0.0.0可能没有意义。第3行拒绝子网172.22.30.0上所有其他主机，这也是相当直观的。但第4行的目的就不是很明显了，它允许地址范围是172.22.0.1~172.22.31.255的主机通过，反码指定了本行的地址范围。第5行拒绝B类网络172.22.0.0的所有子网，最后一行允许所有其他地址。

另一种方法也可以配置相同的列表，参见示例B-7所示。

示例B-7 在这里使用访问列表配置模式也定义了示例B-6中显示的标准IP访问列表

img687_3

开始3个列表条目的序列号是指定的，第4、第5和第6个的条目是分别由其前面的条目加10自动分配的，也就是30、40和50。通过在访问列表中期望位置的上面和下面条目的序列号之间指定一个序列号，就可以简单地在两个条目之间增加一条新的语句。示例B-8就显示了这样的例子。

示例B-8 利用序列号在一个标准的IP访问列表之间增加一条新的语句

img687_4

参见示例B-8，在允许172.22.30.95通过的条目之后和拒绝子网其他地址

的条目（deny 172.22.30.0 0.0.0.255）之前增加一条新的条目。

这个条目也可以被简单地删除。示例B-9中删除序列号为17的条目。

示例B-9 利用序列号，在一个标准的**IP**访问列表中间删除一个条目

img687_5

为了使访问列表在以后更加容易理解，可以在任何条目之前或之后增加一个注释。在示例B-10和示例B-11中的访问列表配置中包括了注释。

示例B-10 在一个标准的**IP**访问列表中增加注释

img688_1

示例B-11 使用路由器的访问列表配置模式可以在一个标准的**IP**访问列表中增加注释

img688_2

示例B-10和示例B-11中演示了两种配置同一个列表的方法。其中，注释对表的功能没有任何影响，但它们可以使一个复杂的访问列表对以后的读者变得更加友好一些。

为了完全理解访问列表的功能，读者需要理解反向掩码

回忆一下IP地址掩码的作用：为了从主机地址导出网络地址或子网地址，掩码中的1对应着网络位，0对应着主机位。在每一位上执行布尔与（Boolean AND）操作可以得到网络或子网号。图B-5（a）包括了“与”（AND）函数的真值表和用英文表达的函数状态。

比较两位，当且仅当两位都为1时结果为1。

img688_3

图B-5 真值表、布尔与和布尔或的例子

布尔或（Boolean OR）是布尔与（Boolean AND）的反函数，真值表如图B-5（b）所示。

比较两位，当且仅当两位都为0时结果为0。

一个反码（inverse mask）（Cisco更喜欢用术语通配掩码（wildcard mask））把对应于地

址位中将要被准确匹配的位设置为0，其他位设置为1——值为1的位经常叫做不关心位。接着反码将同地址进行或操作。

注意，图B-5（b）中的或操作结果是172.22.30.255。在IP术语中这一结果意指子网172.22.30.0上的所有主机地址。来自172.22.30.0的任何指定地址都将匹配到该地址/反码组合上。

图B-6给出了两种书写标准IP访问列表的快捷方法。图B-6（a）给出的全0反码指明被讨论地址的所有32位都必须准确匹配到172.22.30.6。对标准IP访问列表来说，缺省掩码是0.0.0.0。所以给出的替换语句除了不带指定掩码外，同第一个语句完全相同。注意，这个缺省掩码不能应用到下一节将要讨论的扩展IP访问列表。

图B-6（b）给出了允许一切的地址/反码组合。地址0.0.0.0实际上仅是一个占位符，真正起作用的是掩码255.255.255.255。通过将每一位都设置为1，该掩码将匹配任何地址。可以替换该语句的方法是使用关键字**any**，它与第一个语句的意思相同。

img689_1

图B-6 在配置标准**IP**访问列表时可以使用的两种快捷方法

B.3 扩展IP访问列表

扩展IP访问列表在指定过滤内容方面提供了更多的灵活性。扩展IP访问列表的基本格式如下：

```
img689_2
```

读者可以使用和配置标准列表相同的方法，在全局访问列表配置模式下配置扩展的访问列表。

在扩展的访问列表中也可以使用序列号。用法和标准列表中的用法相同。自反访问列表（**reflexive access list**）只可以在全局访问列表配置模式下使用，并且只能配置在命名的IP访问列表中。自反访问列表将在这个附录的后面章节讲述。

这里有一些特性已经很熟悉了，但还有一些新的特性：

- 对于扩展IP访问列表来说，*access-list-umber* 范围在100～199，或2000～2699之间。
- **dynamic** 表示这个列表是一个动态的访问列表。动态访问列表用于“Lock-and-Key”的安全特性。某个用户可以通过Telnet访问一台路由器，路由器使用像TACACS+或RADIUS这样的认证服务器进行认证，并且在动态入口处基于源和目的信息允许或拒绝该用户访问。
- **timeout** 定义了一个临时条目在一个动态列表中保留的最大时间，以分钟计。缺省情况下条目没有超时配置，它永远保留在列表中。
- *protocol* 是一个新变量，它可以在IP包头的协议字段寻找匹配，可选择的关键字是**eigrp**、**gre**、**icmp**、**igmp**、**igrp**、**ip**、**ipinip**、

nos、**ospf**、**tcp** 和 **udp** 。还可以使用0~255中的一个整数表示IP协议号。**ip** 是一个通用关键字，它可以匹配任意和所有的IP协议，同样地，反码255.255.255.255将可以匹配所有地址。

- 注意为了进行匹配，数据包的**source** 地址和**destination** 地址都将被检查；它们有各自的反码。

- **precedence** 和 **tos** 是可选变量，它们可以在IP包头中的优先级字段和服务类型字段寻找匹配。优先级取值范围在0~7之间，TOS在0~15之间，或者用关键字表示，可参见Cisco可用关键字列表文档。

- **log** 是一个可选项，它指定打开信息日志功能。路由器试图记录的信息可能包括匹配的列表号与名字、源和目的地址、上层端口号和匹配记录的数据包数目。

- **log-input** 增加输入接口和源MAC地址或虚电路号到日志输出。

- **time-range** 可以创建一个临时的访问列表。Time-range定义了访问列表有效的时间间隔。扩展访问列表中的**time-range** 参数参考全局**time-range** 命令。全局命令**time-range** 定义了实际的时间参数。

- **fragments** 定义了如何通过访问列表条目处理分段的数据包。根据是否在访问列表中指定第3层或第3层和第4层信息，以及列表的条目是否允许数据包通过来以不同的方式处理分段。缺省情况下（不指定关键字**fragments**），包含第3层信息（IP地址，IP端口号）的条目应用于所有未分段的数据包、初始分段和非初始分段的数据包。对于包含第3层和第4层（除了IP地址还有TCP或UDP端口号）信息的条目，则应用于未分段和初始分段的数据包。这个条目也可以在以下情况应用于非初始分段的数据包：如果非初始分段数据包的第3层信息匹配某个条目的第3层信息（IP地址，IP端口号），并且是匹配允许通过语句，那么这个分段就被允许通过。如果该条目是匹配拒绝通过的语句，那么将处理下一个访问列表。如果指定了**fragment**，条目将仅仅应用于非初始分段数据包。**fragment** 关键字不能配置到包含第4层信息的条目，例如TCP或UDP端口号。

示例B-12演示了一个扩展IP访问列表的例子。

示例B-12 扩展IP访问列表可以通过多种方式允许和拒绝数据包

img690_1

以下对示例B-12中每一行进行说明：

- 第1行：在一个指定的时间范围内，源地址是172.22.30.6且目的地址属于网络10.0.0.0的数据包被允许通过。第7行和第8行定义了时间范围。在其他的时间，这个访问列表条目是没有激活的。没有激活的条目意味着这个条目就像根本没有在访问列表中一样被忽略掉。
- 第2行：源地址是172.22.30.95且目的地址属于子网10.11.12.0/24的数据包被允许通过。
- 第3行：源地址属于子网172.22.30.0/24且目的地址是192.168.18.27的数据包被拒绝通过。
- 第4行：源地址在172.22. 0.0～172.22.31.255之间且目的地址属于网络192.168.18.0的数据包被允许通过。
- 第5行：源地址属于网络172.22.0.0且目的地址前26位是192.168.18.64的数据包被拒绝通过。
- 第6行：从任意源地址到任意目的地址的IP数据包被允许通过。

- 第7行和第8行：命名为“morning”的时间范围定义为每周日早上8:00～11:59，由列表的第1行引用。

图B-7给出了两种书写扩展IP访问列表的快捷方法。回想一下标准IP访问列表曾使用过缺省掩码0.0.0.0，但是这个缺省掩码不能用于扩展IP访问列表，因为路由器无法正确地进行解释。但是对扩展列表也存在一个替代表述。在图B-7（a）中，如果数据包的源地址是主机172.22.30.6且目的地址为主机10.20.30.40，那么数据包被允许通过。在扩展IP访问列表中出现的掩码0.0.0.0，可以通过在地址之前添加关键字host来代替。

img691_1

图B-7 书写扩展IP访问列表的两种快捷方法

在图B-7（b）的例子中，允许从任意源点到任意目标的数据包通过。正像标准访问列表一样，对源地址、目的地址或源和目的地址都可以使用关键字**any** 代替地址/反码组合0.0.0.0255.255.255.255。

扩展访问列表比标准访问列表更强大，因为前者不仅检查数据包的源地址，而且检查所有有价值的内容，但是任何事情都是有代价的。使用扩展列表所要付出的代价是增加处理负担（如图B-8所示）。因为访问列表中每一行都需要检查数据包中的多个字段，因而会发生多次CPU中断。如果访问列表非常庞大或路由器很忙，那么这个要求可能会对性能产生不利的影响。

尽可能保持访问列表的长度较小，这样可以减轻路由器的处理负担。而且还要注意，在匹配发生时，指定的动作会被调用，这时处理会停止。因此，如果你能够使大多数匹配都发生在你所给出访问列表的前几行，那么性能将会被提升。虽然这种方式并不总是可行的，但是在设计访问列表时需要记住这一点。

img692_1

图B-8 访问列表的决策流程图

某些路由器平台支持一个叫做“Turbo ACL”的功能特性，它可以对访问列表进行编译。路由器可以把所配置的访问列表编译到一个查询表中。条目的顺序保持不变，查询时间和查询占用的CPU可以大大降低。像时间范围等一些条目是不能包含在一个编译的列表中的。输入命令**access-list compiled**可以在路由器上进行增强的访问列表配置。

作为练习，请根据示例B-12给出一个更精彩的访问列表配置。也就是说，用尽可能少的行数重写访问列表，但又不损失任何功能（提示：一个相同功能的访问列表仅需要4行，不包括末尾的两个时间命令）。下面一段是答案，在进一步阅读之前请读者尝试重写这个列表。

第1行可以删除。第1行在周日的早晨允许主机172.22.30.6访问地址

10.0.0.0/8。没有这一行，通过第6行仍然可以允许从这台主机到地址10.0.0.0的访问，第6行允许所有没有在它之前被拒绝的数据包通过。

第2行也可以删除。在第6行也允许主机172.22.30.95到10.11.12.0/24的访问。

读者可能在想第4行是否也可以删除，但是请注意第5行拒绝了比包含第4行允许的地址范围更大的地址段。因此，在第5行指定的其余地址被丢弃前，利用第4行允许一个较小的子网地址通过是必要的。

B.3.1 TCP访问列表

检查TCP段的扩展访问列表的行语法如下：

img693_1

注意，协议变量是**tcp**。这里最重要的特性是，访问列表可以检查TCP段头中的源和目的端口号。其结果是，过滤数据包的选项不仅是去往或来自一个特殊地址，而且还可以是去往或来自一个特殊套接字（一个IP地址/应用端口的组合）。

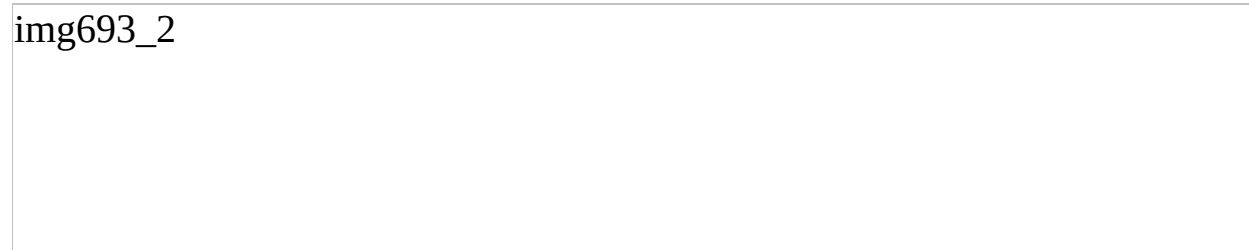
还没有解释的TCP访问列表的特性是**operator** 和 **port**：

- **operator** ——指定逻辑操作。选项可以是**eq**（等于）、**neq**（不等于）、**gt**（大于）、**lt**（小于）和**range**（指明包括的端口范围）。如果使用**range** 运算符，那么要指定两个端口号。
- **port** ——指明被匹配的应用层端口号。几个常用的端口号是Telnet（23）、FTP（20和21）、SMTP（25）和SNMP（169）。完整的TCP端口号列表参见RFC 1700。

假设你实现了一个访问列表可以阻止外部发起的TCP会话进入到你的网络中，但是你又想让内部发起的TCP会话的响应通过，那应该怎么办？通过检查TCP段头内的ACK和RST标记，关键字**established** 可以实现这一点。如果这两个标记都没有被设置，表明源点正在向目标建立TCP连接，那么匹配不会发生。最终数据包将会在访问列表的后继行中被拒绝。

示例B-13显示了一个TCP访问列表的例子。

示例B-13 这个**TCP**访问列表允许已经建立的连接和允许访问某些地址的**SMTP**和**Telnet**端口



img693_2

以下是对示例B-13中的每一行的解释：

第1行：如果连接是从网络172.22.0.0发起的，那么允许从任意源点到该网络的TCP数据包通过。

第2行：允许来自任意源点，且目标端口号是主机172.22.16.83的端口25（SMTP）的TCP数据包通过。

第3行：允许来自网络10.0.0.0，去往网络172.22.114.0/24且目标端口为23（telnet）的TCP数据包通过。

隐式拒绝一切的所有数据包被丢弃。

B.3.2 UDP访问列表

检查UDP段的扩展访问列表的行语法如下：

img693_3

除了协议变量是**udp** 外，该格式与TCP的格式非常相似。另一个不同之处是没有关键字**established**。原因是UDP提供无连接传输服务，在主机之间没有建立连接。

通过向前面例子中添加额外的3行可以得到示例B-14中的例子。

示例**B-14** 这个访问列表允许**TCP**和**UDP**数据包通过

img694_1

第4行：允许从子网10.64.32.0/24到主机172.22.15.87且目标端口为69（TFTP）的UDP数据包通过。


第5行：允许从任意源点到主机172.22.15.85且目标端口为53（域名服务器）的UDP数据包通过。

第6行：允许从任意源点到任意目标的SNMP数据包通过。

在列表中没有发现匹配，隐式拒绝一切停止丢弃所有数据包。

B.3.3 ICMP访问列表

检查ICMP数据包的扩展访问列表的行语法如下：



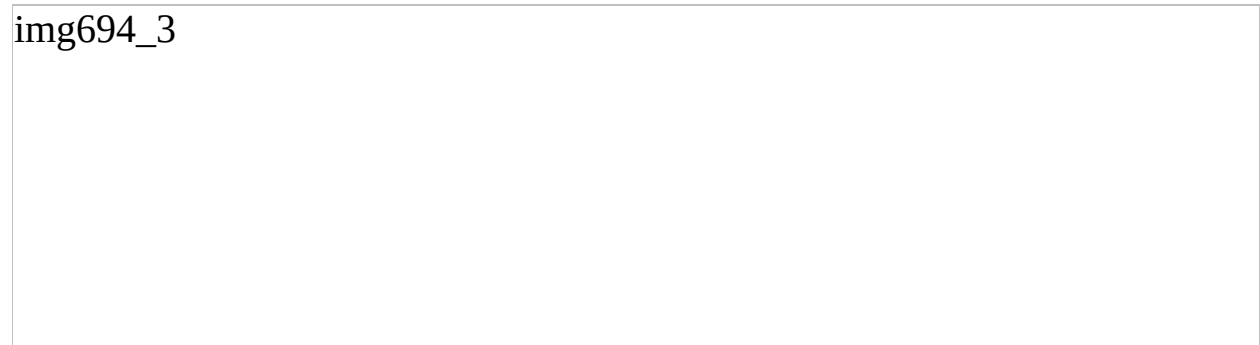
img694_2

icmp 目前在协议字段，注意这里没有源端口或目的端口，因为ICMP是一种网络层协议。该行可以用于过滤所有ICMP消息，或者读者可以使用下面的选项过滤指定的ICMP信息：

- *icmp-type* 编码范围是0～255。所有ICMP类型编号见RFC 1700。
- 过滤粒度可以通过指定*icmp-code* 进行增加。一个ICMP代码指定了ICMP数据包类型的一个子集；这些代码是一个在0～255之间的数字，也在RFC 1700中描述。
- 可以输入ICMP消息的名字替代ICMP类型和ICMP代码。

示例B-15中显示了一个ICMP访问列表的例子。

示例B-15 这个**ICMP**访问列表拒绝指定的数据包和允许其他所有的数据包



img694_3

以下是对示例B-15中每一行的解释：

第1行：拒绝从网络172.22.0.0到任意目标的ICMP ping响应（回应应答，ICMP类型0）通过。

第2行：拒绝从网络172.22.0.0到任意目标的ICMP目的网络不可达数据包通过，其中代码号为9。

第3行：拒绝从网络172.22.0.0到任意目标的ICMP目的网络不可达数据包通过，其中代码号为10。

第4行：拒绝从网络172.22.0.0到任意目的地的ICMP traceroute。

第5行：允许所有其他IP数据包通过。

B.4 调用访问列表

如果不通过调用命令使数据包发送到访问列表，访问列表是不进行任何处理的；这里所调用的命令定义了如何使用访问列表。命令如下所示：

ip access-group *access-list-number* {**in**|**out** }

在接口上配置这条命令可以建立安全过滤器或流量过滤器，并且可以应用于进、出流量。如果**in** 或**out** 关键字都没有被指定，那么缺省值是出站。当然，访问列表编号指定了接收该命令所发送数据包的访问列表。图B-9给出了该命令的两种配置。

img695_1

图B-9 命令**ip access-group**使用指定的访问列表在接口上针对进出流量建立过滤器

图B-9中的访问列表1过滤进入接口E0的IP数据包，它对于出站数据包和其他协议（如IPX）产生的数据包不起作用。访问列表2过滤离开接口S3的IP数据包，它对于入站数据包和其他协议产生的数据包不起作用。在入站访问列表中尽可能的执行拒绝语句来代替出站列表，可以减少路由器处理那些将要被丢弃的数据包所花费的系统资源。

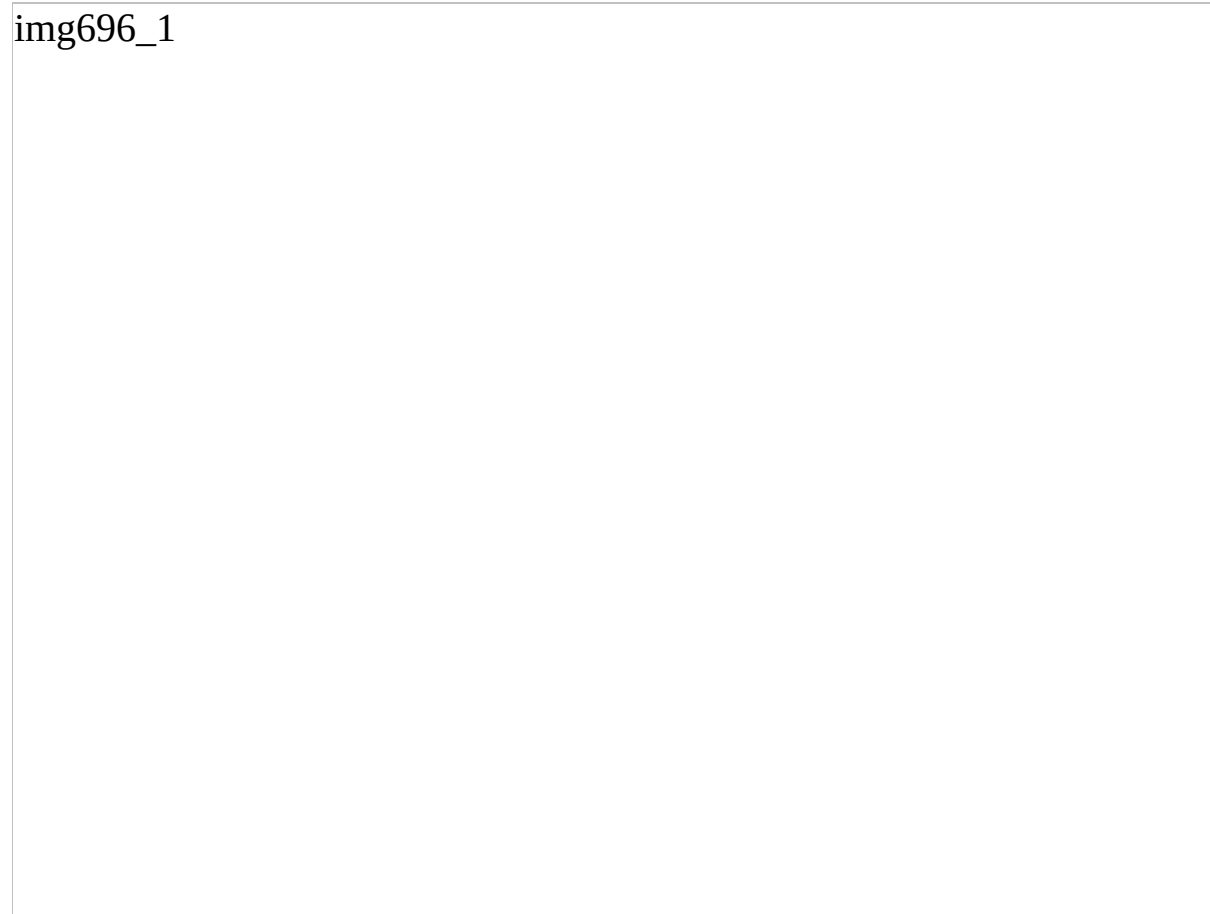
多个接口可以调用相同的访问列表，但是在任意一个接口上，对每一种协议仅能有一个进入和离开的访问列表。

在图B-10中，前面例子中给出的TCP、UDP和ICMP访问列表被用作过滤器。源自前面两个例子中的访问列表110被应用到以太网接口0上，用来检查入站流量。应用到相同接口的访问列表111检查出站流量。仔细分析一下两个访问列表，包括它们之间的相互关系，再考虑下面问题：

- 从172.23.12.5到10.64.32.7的ping响应想从接口Ethernet0出站，是否允许通过？
- 想在主机172.22.67.4上ping网络10.64.32.20上的一台设备，从接口Ethernet0出站可以ping通吗？

来自172.23.12.5的ping响应包允许从接口Ethernet0出去。来自172.22.0.0/16的ping响应包会被拒绝，而不是172.23.0.0/16。从172.22.67.4到10.64.32.20的ping包，从接口Ethernet0出去将不被允许。这个ping请求可以成功地从该接口出去，但它的响应包将被入站访问列表拒绝。

img696_1



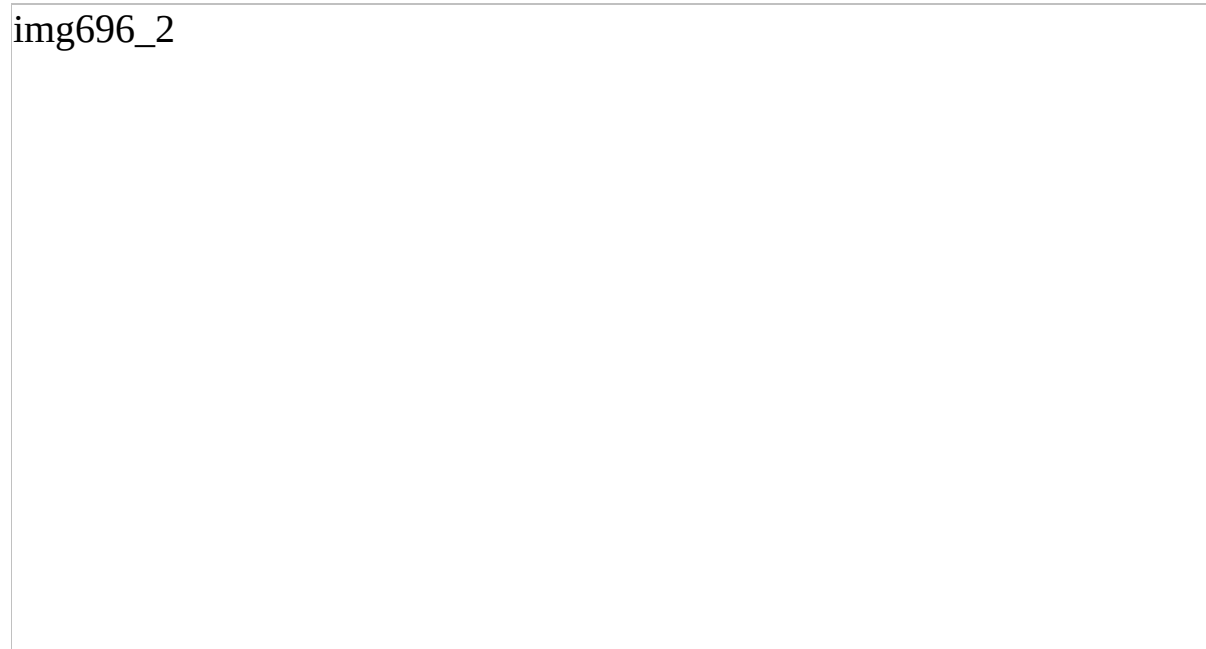
图B-10 这里访问列表**110**用于过滤以太网接口上的入站数据包。访问列表**111**用于过滤该接口上的出站数据包

另一种调用访问列表的命令是**access-class**。该命令用于控制到达路由器或由路由器虚拟终端线路发起的telnet会话，而不进行数据包过滤。命令格式如下：

```
access-class access-list-number {in|out }
```

图B-11给出了使用命令**access-class**的例子，访问列表3控制路由器VTY线路将要接受的telnet会话的源点地址。访问列表4控制路由器VTY线路可以连接的目标地址。

img696_2



图B-11 命令**access-class**使用访问列表控制到达和由路由器虚拟终端线路发起的telnet流量

命令**access-class** 对于路由器传输的telnet流量不起作用，它仅影响到达路由器以及由路由器发起的telnet会话。

B.5 自反访问列表

自反访问列表是自动驻留的、暂时的、基于会话的过滤器。如果某台路由器允许通过网络内部向外部的主机初始发起一个会话，那么自反列表就允许返回的会话数据流。自反列表和被扩展命名的IPv4访问列表一起使用。使用自反列表的会话过滤器可以与使用关键字**established**的TCP过滤器进行比较。使用**established**关键字，TCP会话是从网络内部初始发起的。如果返回的数据流具有ACK或RST的设置，那么该数据包就是以前建立连接的会话的一部分，并且该数据包被允许通过。带有**established**关键字的条目是访问列表中的永久条目。

自反访问列表使用不同的参数来确定数据包是否是以前建立的会话的一部分。对于TCP或UDP数据包来说，自反访问列表使用源和目的IP地址，以及源和目的TCP或UDP端口号。

当存在一个从网络内部初始发起的会话时，自反访问列表就会保留从初始的数据包中收集的会话信息。反转并添加源和目的IP地址以及源和目的端口号，连同上层协议类型（例如TCP和UDP），作为临时自反列表的允许语句。该条目在出现以下几种情况之前都是保持活动的：不再有任何有关该会话的数据流和超时值；收到两个FIN标记的数据包；或者在TCP数据包中设置了RST标记。

示例B-16显示了一个自反访问列表配置的例子。

示例B-16 示例中的自反访问列表命名为**infilter**

img697_1

在这个示例中，在与外部网络连接的接口上应用了过滤器。outfilter列表只允许在每周日早晨9:00~12:30之间，从内部网络初始发起的所有TCP数据包和ICMP echo请求数据包通过。在串行接口Serial0/0.1的出站方向应用了outfilter列表。在permit语句中使用了关键字**reflect**，它创建了一个名为“sessiontraffic”的自反访问列表。在出现匹配这个带有**reflect**关键字的permit条目时，就会保留这个自反访问列表。

即将进入接口Serial0/0.1的数据包需要通过infilter访问列表进行过滤。这些数据包是源自某个外部网络的。在这个实例中，infilter列表允许EIGRP和RIP的数据包通过。当进入的数据包匹配EIGRP和RIP条目后，自反访问列表sessiontraffic将依次被判定。自反访问列表在末尾没有隐含的deny-all语句，但包含自反访问列表的扩展访问列表末尾具有该语句。

示例B-17显示了在TCP和ICMP流量从接口Serial0/0.1出去之前的访问列表。

示例B-18显示了在TCP和ICMP流量从接口Serial0/0.1出去之后的访问列表。

示例**B-17** 命令**show ip access-list**显示了路由器上配置的所有永久与临时的**IP**访问列表

img698_1

从内部网络向外部网络发起ping和telnet操作。

示例B-18中显示了发起上述流量后的访问列表信息。

示例**B-18** 命令**show ip access-list**显示了路由器上动态创建的自反访问列表的条目

img698_2

示例B-17的输出信息显示了访问列表进站和出站过滤，以及它们配置参数。请注意ICMP条目在出站过滤中是active的，这意味着现在路由器的时间和这周的日期落在所配置的时间范围之内。输出信息也显示了没有驻留的自反访问列表sessiontraffic。

在发起ICMP的ping和telnet会话，数据包通过serial0/0.1出去后，在示例B-18中再次显示了访问列表信息。这次就有了自反访问列表的条目。这些条目是匹配所有从外部网络到达serial0/0.1的数据包的，一直持续到计时器超时或会话被关闭。从外部网络发起telnet会话和ICMP echo不会成功。

自反访问列表不支持那些在会话期间改变端口号的协议，例如FTP。

B.6 可供选择的關鍵字

大多数网络专业人员都知道一些较常用的TCP端口号和一些UDP端口号。很少有人可以说出有关ping或目标不可达的ICMP类型是什么，知道目标不可达类型的ICMP代码的人就更少了。从IOS 10.3开始，配置访问列表可以使用关键字来代替端口、类型或代码编号。使用关键字，访问列表110和111显示在示例B-19中。

示例B-19 在访问列表中关键字替代了端口号

img699_1

注意，如果你将路由器从10.3以前的版本升级到新版本，紧接着重启路由器，那么路由器将会使用新的语法重写配置文件中的访问列表，包括关键字。如果你随后又需要重载最初10.3之前的镜像文件，那么路由器将无法理解被修改的访问列表。记住在任何情况下，升级之前要将原来的配置文件上载到TFTP服务器上。

B.7 命名访问列表

对于每台路由器，798个标准访问列表或799个扩展访问列表的限制看上去已经够用了，但是有些情况（比如动态访问列表[\[1\]](#)）可能就不够了。从IOS 11.2开始提供的命名访问列表打破了这种限制，它的另一个优点是描述名可以使数量庞大的访问列表更加便于管理。

为了使用命名访问列表，访问列表的第1行应采用以下格式：

```
ip access-list {standard|extended } name
```

因为没有编号区分访问列表类型，所以该行可以将列表指定为标准IP列表或扩展IP列表。下面紧接着可以使用许可或拒绝语句，标准列表的语法如下：

```
{deny|permit } source [source-wildcard]
```

基础扩展列表的语法如下：

```
img699_2
```

在这两种情况下，都没有出现命令的**access-list access-list-number** 部分，但是其他部分完全相同。在相同路由器上标准和扩展访问列表不能同名。除了在接口上建立命名访问列表的命令涉及到用名字替代编号外，在其他方式中命令均保持不变。图B-12使用命名格式对图B-10中的访问列表进行了转换。

img700_1



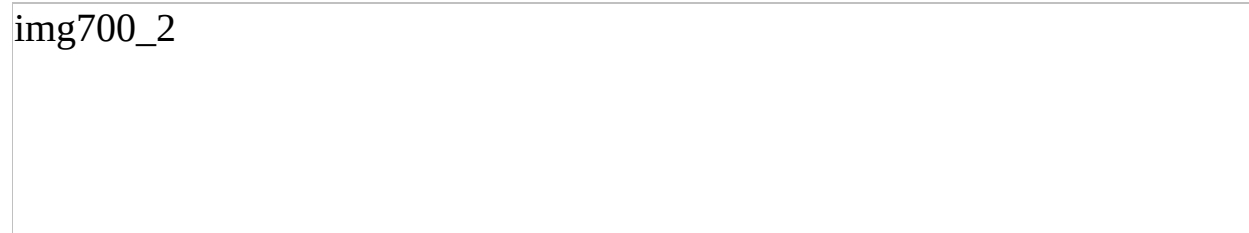
图B-12 图B-10中的访问列表现在被配置为命名访问列表

B.8 前缀列表

前缀列表（prefix list）用于在路由更新时允许或拒绝指定的某个地址或地址范围。BGP路由选择协议在IPv4中使用前缀列表。在IPv6的协议之间交换IPv6地址或过滤更新时，所有IPv6的路由选择协议都可以使用前缀列表。

前缀列表可以被命名。表中的条目允许或拒绝通过一个地址或地址范围，参见示例B-20。

示例B-20 前缀列表允许和拒绝IPv6地址通过



img700_2

第1条允许长度精确为64位的前缀2001:db8:0:1::通过。第2条和第3条允许通过一段地址范围。关键字**le**表示一个前缀长度范围，它所匹配的前缀长度大于等于该语句中的前缀后面指定的长度，小于关键字**le**后面指定的长度。前缀列表v6_addr_filt的第2条允许匹配2001:db8:0:10::和长度在60~64之间的前缀。关键字**ge**指定了某个地址范围内的前缀最小长度。如果没有包含关键字**le**，它就假定所匹配的前缀范围的最大长度为128位，即IPv6前缀的最大位数。当包括关键字**le**时，地址范围的最大匹配长度就是**le**后面指定的长度。前缀列表v6_addr_filt的第3条允许所有长度在62~64之间的前缀通过。

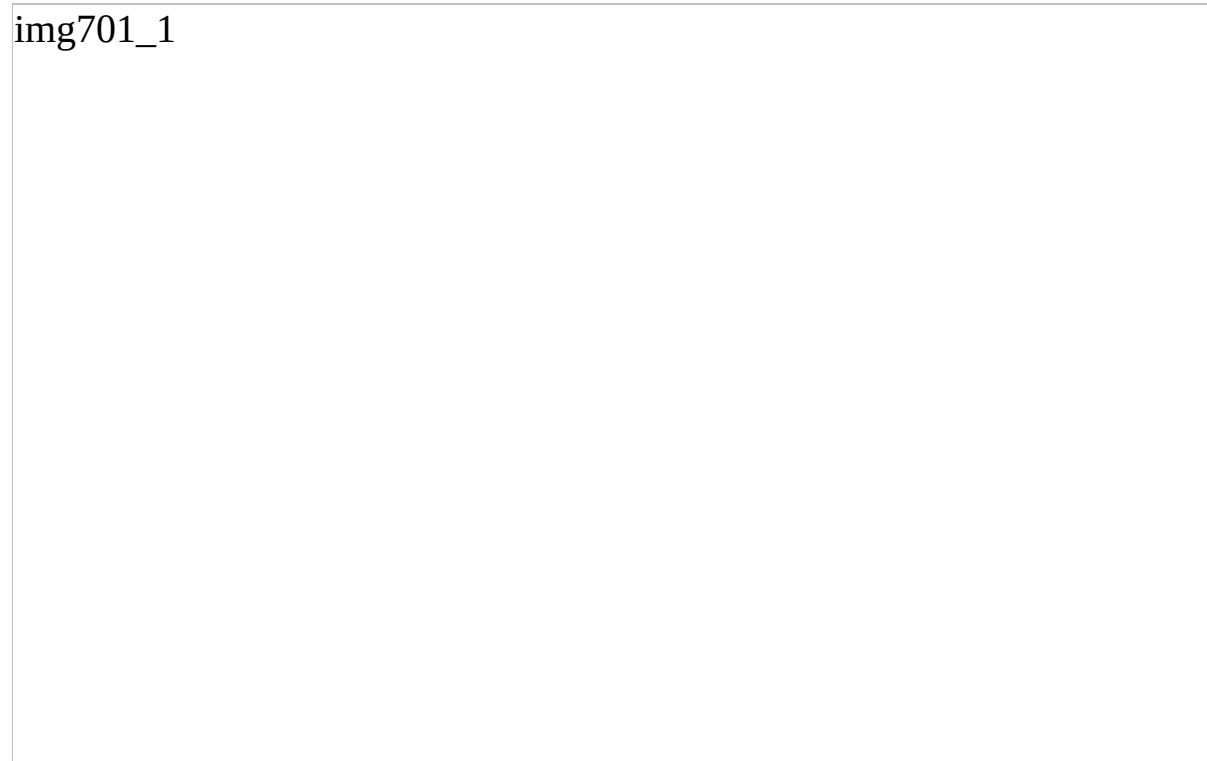
B.9 对放置过滤器的考虑

为了达到最佳的性能，你不仅需要考虑访问列表自身的有效设计，而且还要考虑如何在路由器和网络中去布置过滤器。

凭借经验，安全过滤器通常是作为入站过滤器。在不必要或不被信任的数据包到达路由选择进程之前将它们过滤掉，阻止欺骗攻击——数据包欺骗路由选择进程，使其认为它来自某处，但它实际并不是来自那里。另外一方面，流量过滤器通常是出站过滤器。当你考虑在某一点的流量过滤器可以阻止不必要的数据包占用某条数据链路时，这种方法将很有意义。

除了这两种经验外，要考虑的另一个因素是访问列表和路由选择进程将要使用的CPU周期数。入站过滤器是在路由选择进程之前被调用，而出站过滤器是在路由选择进程之后被调用（如图B-13所示）。如果大多数经过路由选择进程的数据包被访问列表拒绝，那么入站过滤器可以节省一些处理周期。

img701_1



图B-13 入站过滤器是在路由选择进程之前被调用，而出站过滤器是

在路由选择进程之后被调用

标准IP访问列表仅能过滤源地址。因而，使用标准列表的过滤器必须被放置在尽可能靠近目标的地方，以便源点还可以访问到其他没有被过滤的目标（如图B-14（a）所示）。其结果是带宽和CPU周期被浪费在最终将要被丢弃的数据包上。

因为扩展IP访问列表可以非常准确地标识数据包特性，所以它们应该被放置在尽可能靠近源点的地方，这样可以防止带宽和CPU浪费在传输无用的数据包上（如图B-14（b）所示）。另一方面，扩展列表的复杂性意味着更多的处理负担。当决定在那里放置过滤器时，需要权衡考虑。

你还必须理解访问列表将怎样影响路由器上的交换。例如，使用扩展访问列表的接口不能被独立地交换；动态访问列表不能被硅交换，而且可能影响硅交换的性能。在IOS 11.2版要之前根本不支持命名访问列表。

在骨干或核心路由器上，访问列表对交换的影响可能是很重要的。通过阅读Cisco关于IOS的配置指南可以确保全面地研究和理解访问列表所产生的影响。在某些情况下，一个数据包过滤路由器——一个专门用于数据包过滤的较小的路由器——可能被用来减轻重要路由器的负担。

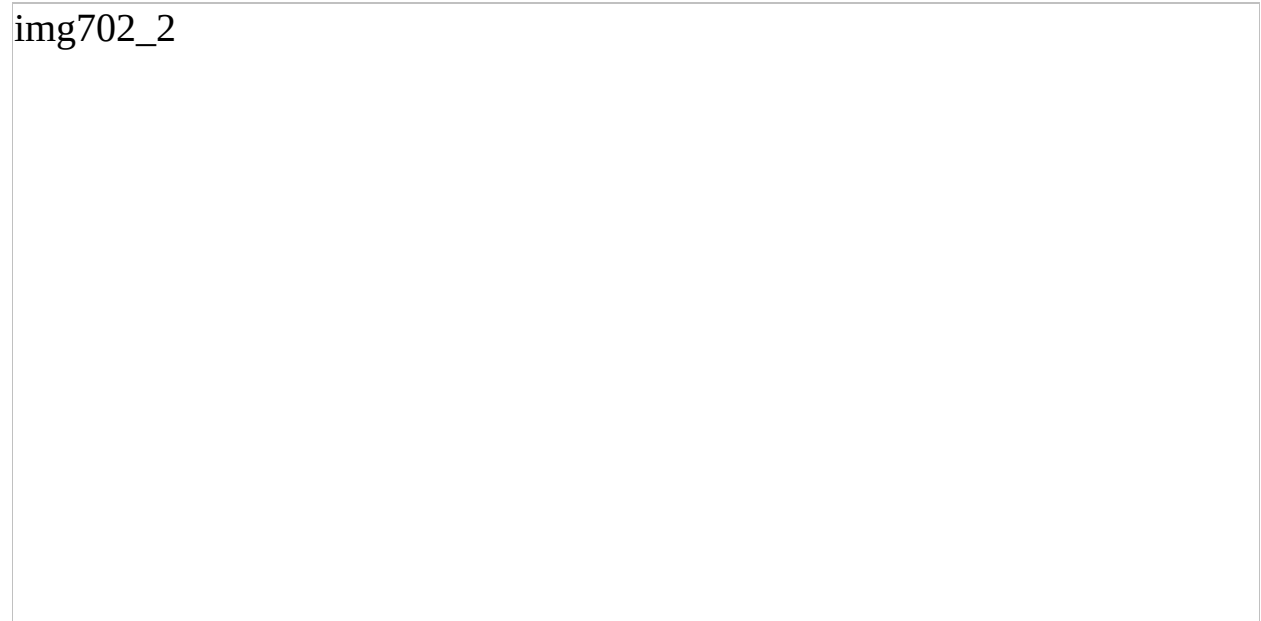
img702_1

图B-14 使用标准列表的过滤器必须被放置在尽可能靠近目标的地方（**a**），而扩展访问列表可以放置在靠近源点的地方（**b**）

B.10 访问列表的监视和计费

在不必显示路由器的完整配置的前提下，能够检查一个甚至所有访问列表是非常有用的。命令**show ip access-list**可以显示路由器上所有IP访问列表的简化语法。如果要观察一个特定的访问列表，可以通过指定名字和标号（参见示例B-21）。如果没有输入关键字**ip**（**show access-list**），那么所有访问列表都将被显示出来。

示例B-21 命令**show ip access-list**可以显示访问列表的简化语法



img702_2

跟踪被访问列表拒绝的数据包作为安全计划或编制策略能力的一部分，这是很有用的。命令**ip accounting access-violations**可以配置在独立的接口上，用来创建所有在该接口被访问列表拒绝的数据包的数据库。可以使用命令**show ip accounting access-violations**来检查数据库，结果可以显示出匹配该地址的源和目标地址、数据包数、字节数以及拒绝该数据包的访问列表编号（参见示例B-22）。命令**clear ip accounting**可以清除计费数据库。

示例B-22 访问列表计费数据库可以使用命令**show ip accounting access-violations**来查看

img703_1

计费功能将关闭接口上的独立交换功能和硅交换功能。在需要这些交换模式的接口上不能使用计费功能。

作为最后一种技巧，你应该知道被列表尾部隐含地拒绝一切所丢弃的数据包，它们是不能被计费跟踪的。为了跟踪这些数据包，仅需在列表尾部配置一条拒绝一切的条目，参见示例B-23。

示例B-23 在访问列表的尾部增加一条**deny any**条目，用来跟踪因为不匹配列表中所有其他条目的而被丢弃的数据包

img703_2

[\[1\]](#) 本教程中没有涉及动态访问列表。详细内容可参见Cisco IOS安全配置手册——配置Lock-and-Key Security（动态访问列表），查询更多的信息。

附录C

CCIE备考提示

要成为Cisco认证互连网络专家（CCIE），远远不像业界其他一些认证，只是“读一本书，参加一次考试”就行了。在通过了相对简单的笔试后，你需要在一场极其艰难的实验室实践考试中展示你的专业才能。虽然你要对Cisco的配置命令必须非常熟悉，但在实验室考试中最大的挑战是与Cisco IOS软件无关，他们测试的是你对交换机、路由器以及路由选择协议的理解程度。正因为如此，CCIE们才被认为是千里挑一的互连网络专家。

有层次地构建网络包括4个步骤；这4个步骤同样也适用在准备CCIE实验室考试的过程中。

- 计划：冷静而正确地评估你现在的经验和不足之处。计算一下你每天能用来学习的时间。检查你的资源，包括实验室设备、资金、书籍、训练的时间以及能请教问题的熟人、老师、专家。衡量你自身的长处和短处：你擅于考试吗？你在压力下能很好地工作吗？你对挫折和失落的反应是什么？你学习习惯好吗？你能很好地通过阅读来学习吗？通过利用以上的答案，编写一张计划表，按要求去执行你的计划。
- 设计：尽可能多地与CCIE交流，询问他们的准备方案，并按你的要求，扬长避短地设计你的准备方案。你的方案应该能准确地反映出在一定的时段内将你现有水平提升至CCIE Lab的水平。你还可以通过一系列具有明确目标的小工程来架构一个大的项目。综合你各方面的因素，来制订你的计划，否则你的计划会受到很大影响。
- 实现：有很多失败的例子是因为在设计完成之前就忙于实施。应该清楚地编写你的准备计划，规划好方案的每一步，而不要草率实施。一旦你开始实施了，就要坚持不懈。不要放弃，也不要气馁，更不要懒惰。检查核对你要完成的每一步。
- 优化：你的准备计划应该是一个生动的文档。当你每前进一步

时，都会遇到比你预计的要么困难要么简单的问题。你可以采用灵活的态度，增加额外的任务来帮助你掌握每一个主题。

只有你才能设计最适合自己的准备计划。在下面的几节里，我给出的建议并不是叫你死板地遵循，而是给你一些点子，让你创建自己的学习计划。一些小的技巧来源于我作为一个CCIE和Cisco系统讲师的经验，也来源于我的那些成功通过CCIE考试的同事们的经验。

C.1 牢固的基础

如果你是一名初学者，或者你的连网经验有限，你的第一步就是牢牢掌握网络互连和Cisco路由器的基础部分。这个过程包括课堂训练和自学。我建议参考Cisco.com，可以在“Learning and Events”中查找有关职业认证的有用信息。

Cisco通过它的培训伙伴提供了很多实践培训课程。只要你的时间和资源允许，尽可能多地参加这些培训。其中一些培训特别重要：

- Cisco网络设备互连（ICND）；
- 组建可扩展的Cisco互连网络（BSCI）；
- 组建Cisco远程访问网络（BCRAN）；
- 组建Cisco多层交换网络（BCMSN）；
- Cisco网络故障诊断（CIT）。

充分利用你参加的每一节课，多问老师问题，多和同学们一起讨论。最重要的是，好好利用你接触设备的机会。不要光满足于表面，一定要充分理解“为什么”和“如何做”。当你做完一个实验，不要停止，多掌握一些设备方面的知识，看看有哪些配置和调试命令，多尝试使用它们。如果有时间，多配置几次直到熟练。

上课会让你找出你自身知识的不足。多读书，填补你对网络互连协议和技术方面知识的空白。Internet上有很多好的技术指南，当你开始学习一种新知识时，一定要记得在Web上进行搜索。

C.2 认证途径

在撰写本书第一版的时候，CCIE还是Cisco公司提供的惟一认证。目前，已经具有很多级别的认证了，甚至在CCIE认证项目上也具有了很多专门的认证。制定这些分级认证的好处是可以给你鼓励（虽然它们也不是必须要预先通过的），在你通向CCIE的道路上实现阶段性的目标，也可以在这个过程中被授予具有价值的认证。

认证的第一个目标是Cisco认证网络工程师（CCNA）。这个级别的认证可以证明你掌握了基本的网络互连和网络协议方面的知识，能够安装和实现小型的LAN、WAN和接入网络，同时理解基本的Cisco硬件和IOS软件功能。

Cisco认证的第二个目标是Cisco认证资深网络工程师（CCNP）。这个高级别的认证表明你掌握了大型企业网络运行和故障诊断方面的技术。拥有了CCNP认证资格，你就做好了向专家级别进军的准备，可以开始你的CCIE认证之旅了。

C.3 实践经验

几乎所有的CCIE都会告诉你，实践经验对于准备实验室考试来说是最有价值的。决不要放过任何一个配置和调试路由器的机会。如果你现在的工作无法接触到路由器或者交换机，那么和你们公司的网络工程师打好交道，向他们讲述你的目标，尽可能多地与他们取得联系。

如果你能接触实验室设备，就一定要充分利用这些设备。在实验室取得的经验是不可替代的，在实验室，你可以随心所欲地进行配置，引入各种各样的问题，而不用担心实际环境中的网络崩溃。

有些在线的公司提供远程访问的付费实验室。所付的费用差别很大，这根据你能使用的设备情况而定。

还有一种选择，就是建立自己的实验室。虽然代价昂贵，但作为一名CCIE，你最终所拿的薪水会让你感到自己的投资非常划算。许多旧Cisco设备的价格是很合理的。请订阅Cisco新闻组，comp.dcom.sys.cisco，或groupstudy.com这样的学习组；经常有人在上面卖旧路由器，你或许能与他们进行一次划算的交易。在本书案例中使用的大多数我自己的实验室设备都是在eBay上购买的。虽然两台路由器也可以用，但最好你至少拥有4台路由器，其中一台应该有4个或者更多的串口，这样你可以将它配置为帧中继交换机。记住你不需要顶级设备，过时的路由器对你而言就已经很好了，因为没有人会在一个商用的网络中使用它们。

C.4 深入学习

掌握了一定的基础知识，同时已具有实际动手能力，你应该开始深入学习网络协议了。至少你应该阅读本书中列出的RFC，相关的RFC读得越多越好。www.ietf.org是最好的站点之一。

当然，不是所有的网络协议在RFC中都有描述。寻找更高级的非IP协议方面的教程、白皮书和指南，这些可能会包含在考试中。你也应该学习Ethernet和WAN协议，比如T-1、ISDN、帧中继和ATM。你可以在www.cisco.com上找到非常有价值的公开资料。当然，除了本书外，Cisco Press还提供了大量书籍，如CCNA、CCNP和CCIE方面的书籍，读者可以在Ciscopress.com上去寻找你所需要的书籍。

一个很好地将理论和实践知识结合在一起的学习工具是Cisco新闻组<http://comp.dcom.sys.cisco>在上面可以提交你遇到的具有挑战性的难题，你先进行自我解答，然后等待由CCIE们以及Cisco工程师们给出答案，看看你的答案是不是正确，如果不正确，找出问题所在。你也可以将问题发送到Cisco的新闻组，大多数参与者都比较友好，并愿意分享他们的经验。

还有一些网络上的学习组是专门针对准备CCNP和CCIE考试的。其实历史最久和最著名的一个学习组是groupstudy.com。参加这样的小组是获取资源、分享经验、加入讨论，以及获取疑难问题答案的最佳途径。我强烈推荐读者加入到在线的学习组中去。

最后，如果你找到一些志同道合者，组成一个学习小组吧。我知道某些公司的CCIE学习小组非常有效，从中诞生了很多CCIE。

C.5 最后6个月

具有坚实的理论和实践背景，你最后6个月的准备应该包括仔细阅读“Cisco IOS配置指南”。阅读每一章时，回顾“Cisco IOS命令参考”中的相关章节，确保对于每一个协议，你已经熟悉IOS中的所有配置命令；然后在你的实验室中用不同的方法来配置各章节中所覆盖的协议。尝试不同的假设，并使协议运行在不常见的环境下。你会发现，当一种配置如你所料运行时，你并不能得到最好的经验，在与之相反的情况下，你却往往能学到更多的东西。

为你的配置和想法作笔记，记下它们是如何工作或者不工作的。记得去发掘记录所有与协议有关的调试工具，譬如**debug** 和**show** 命令。

在配置手册每一章的最后，你的目标是应该至少配置协议的最本质部分。当你参加CCIE实验室考试时，不用动太多的脑筋去配置相对简单的部分，抽出时间去思考困难的问题，这是很重要的。你也应该了解一个协议是怎样运作的，协议之间是如何相互影响的，存在哪些配置选项，以及怎样进行协议的故障诊断，等等。当你达到每一章的目标之后，进入下一章。

最后6个月的开始，你应该参加CCIE的笔试。不要被这个笔试所迷惑，它只是想淘汰那些完全没有准备的人，好让他们仅仅损失参加笔试所花的钱，而不是浪费参加实验室考试所需的更大的费用。如果你好好准备了，你会发现其实笔试并不是很难。

一些Cisco培训伙伴提供CCIE的准备课程和“CCIE boot camps”，给你一些在类似CCIE实验室考试的条件下做实验的经验。不要以这些课程取代勤奋的学习和实践。如果你选择参加这些课程，你首先应该认真准备了CCIE考试。从这些商业培训中得到的最大好处就是“在枪口下”的感觉——在紧迫的时间限制中解决困难问题。

C.6 参加考试

CCIE实验室考试不仅仅测试你的实践和理论知识，而且测试你在压力下运用这些知识的能力。既然要参加了这样紧张的一天考试，就不要无谓地为自己再增加压力。

- 大多数人第一次参加CCIE实验室考试都会失败，因此要有参加第二次考试的准备。这能帮助你在参加第一次考试时保持冷静，或许，你就可能一次成功。
- 安排好你的行程，提前到达考场。考试之后再安排回去的行程。你显然不愿意考试期间考虑行程安排。
- 考试的前一天晚上确定考场的位置，不要因为迷了路慌慌张张出现在考场上。
- 考试前一天晚上最多做做小复习，如果你想填鸭式的恶补，只能使自己焦虑不安、失眠。
- 吃一顿好的晚餐，不要喝酒，美美睡一觉。
- 考试当天，吃一顿好的早餐。吃得好有助于你表现更出色，这是事实。
- 穿舒服一点，穿得好不会给你加分。

考试之前，会要求你在线在一张表格上签名，表明你不会泄漏实验室考试题目的细节。考试长达8小时，有一次午餐休息。虽然其他的考生在同一个房间里考试，但你得独立完成。通常，其他考生的题目和你的是不同的。考官会分配给你一个网络拓扑和一组问题，你必须在8小时内完成所要求的配置。

在考试的任何一个部分，如果你不明白某个特定的要求，不要犹豫，去问监考官，他会帮助你。最重要的是，放松并集中注意力。

完成考试就会进行实验室考试评分，结果通常会在考试后的下一个工作日发送给你。你会收到一个通知你考试结果的电子邮件，并可以访问在

线报考页面查看你的成绩。如果通过了考试，你将会看到你的CCIE编号；如果没有，你将会看到一个得分报告。

当你通过CCIE考试，你就完成了值得你万分骄傲的一件事。如果本书或者附录中的一些建议曾经帮助你达成你的目标，请给我发电子邮件，让我分享你的骄傲。

附录D

复习题答案

第1章

1. TCP/IP协议簇的5个层是：

- 物理层；
- 数据链路层；
- IP层；
- 主机到主机层；
- 应用层。

物理层 ——包含关于物理介质的协议。

数据链路层 ——包含了如何控制物理层的一些协议：介质是怎样存取和共享的，介质上的设备是怎样标识的，数据在介质上传播之前是怎样成帧的。

IP层 ——包含一些将数据链路逻辑分组到一个网络以及跨网络进行通信的协议。

主机到主机层 ——包含一些定义并控制中逻辑的、端到端的路径的协议。

应用层 ——对应于OSI中的会话层、表示层以及应用层。

2. 现在使用最普遍的IP版本是版本4。

3. 当数据包的长度超过它所要去的那个数据链路的MTU（Maximum Transmission Unit）时，路由器要将它分段。数据包中的数据将被分成

小段，每一段被封装在独立的数据包中。接收端使用标识符、分段偏移域以及标记域的MF位来进行重组。

4. TTL域防止丢失的数据包在IP网络中无休止地传播。该域包含一个8位整数，此数由产生数据包的主机设定。数据包每经过一台路由器，TTL值将被减1。如果一台路由器将TTL减至0，它将丢弃该数据包并发送一个ICMP超时消息给数据包的源地址。

5. IP地址的分类如下：

- A类：第一个八位组字节的第1位是0；
- B类：第一个八位组字节的前2位是10；
- C类：第一个八位组字节的前3位是110；
- D类：第一个八位组字节的前4位是1110；
- E类：第一个八位组字节的前4位是1111。

6. A、B、C类IP地址用点分十进制以及二进制表示如下：

类	第一个八位组字节 二进制范围	第一个八位组字节 十进制范围
A	00000001~01111110	1~126
B	10000000~10111111	128~191
C	11000000~11011111	192~223

7. IP地址掩码标识了IP地址的网络部分。32位掩码中的1标识了IP地址中相应的网络位，0标识了主机位。将IP地址和掩码进行布尔与（AND）运算，结果是，IP地址中对应于掩码网络部分的那一段不变，而对应于主机部分的全变成0。

8. 子网化是对A、B、C类地址进行子分组。如果没有子网化，A、B、C类的主IP地址的网络部分将只能标识一个数据链路。子网化使用主IP地址中的一些主机位作为网络位，允许一个单独的主地址被划分为多个网络地址。

9. 有类别路由选择协议不能区分全0子网和主IP地址，也不能区分全1

子网和主IP地址的全主机、全子网广播地址。

10. ARP（地址解析协议）的作用是将数据链路上接口的IP地址映射到相应的MAC地址。

11. 代理ARP是IP路由器的功能之一。如果路由器收到一个ARP请求，并且

- 目标网络或子网在路由器的路由表中；
- 路由表指出目标可以通过某一个接口可达，该接口不同于接受到ARP请求的那个接口；
- 路由器将用自己的MAC地址对该ARP请求进行回应。

12. 重定向是IP路由器的功能之一。如果一台设备向原设备发送给路由器一个数据包，而且该路由器必须转发该数据包至同一数据链路上的下一跳路由器，那么该路由器将向原设备发送一条其可以直接到达下一跳路由器的重定向消息。

13. TCP在无连接的IP层之上提供了面向连接的服务。UDP则提供了无连接服务。

14. 序列号确保了准确的排序；校验、确认、计时器以及重传机制确保了可靠性；滑动窗口机制确保了流量控制。

15. MAC地址是固定长度的二进制整数。如果用MAC地址作为IP地址的主机部分，那么子网化将不能得以实现。因为不可能灵活地使用一些主机位作为网络位。

16. UDP报头的惟一目的是增加源端口及目的端口号。

第2章

1. IPv6地址的长度是128位。
2. IPv6地址可以表示为通过冒号分开的8个16位以十六进制表示的分段。
3. 简化IPv6地址的两条规则是：
 - 任何一个16位分段的前导0都可以省略；
 - 由全0构成的一个或多个16位分段的任何单个连续的字符串都可以表示为一个双冒号。
4. 使用多个双冒号会使地址变得含混不清，这样不能准确地确定每一个含0字符串的长度。
5. 两个地址都是全0。::/0表示一个缺省地址，而::/128表示一个未指定的地址。
6. 单播IPv6地址的主机部分是接口ID，长度通常是64位。
7. 单播IPv6地址的子网ID是16位长。
8. 起始于FF80::/10的IPv6地址是一个链路本地地址。
9. 这是一个全球单播地址，开始3位以001标识。
10. 任意播地址是表示一个服务的地址而不是表示一台设备的地址，因此它可以代表多台设备。
11. 多播地址是表示一组设备的地址，不是表示单台设备的地址。
12. IPv6报头的长度为40字节。
13. 流标记字段通过在报头中标记各自不同的流（具有相同的源地址与目的地址和相同的源与目的端口的数据包），允许高颗粒度的负载分

担，而不用因为检查数据包的负荷而降低性能。

14. IPv6下一报头字段相当于IPv4中的协议号字段。命名之所以不同是因为，这个字段的值指定的可能是随后的协议报头，也可能是一个IPv6扩展报头。

15. 跳数限制字段对应于IPv4中的生存时间（TTL）字段。命名改变是因为路由器从来没有根据传送时间来递减这个字段的值，而是在每经过一台传送的路由器时将该字段递减1，实际上就是用跳数替代了传送时间。

16. IPv6下一报头字段和IPv4中的协议号字段一样，也是一个8位字段；如果下一报头是一个上层协议报头，那么它指的就是协议号。但是它也可以指定为与协议号字段不同的字段；如果下一报头是一个IPv6扩展报头，那么它指的就是报头的类型号。

17. 扩展报头使IPv6报头显得更有效率，它可以专门指定专用功能，并只在这个专用功能使用的时候才被包含。

18. ICMPv6（对应于协议号）的下一报头值是58。

19. 除了分段扩展报头外，IPv6分段与IPv4分段的重要不同是，IPv6路由器不对数据包进行分段。它会告诉始发主机要么对数据包进行分段，要么确保不发起太大的数据包。

20. NDP使用的5个ICMPv6消息是路由器请求（RS）、路由器通告（RA）、邻居请求（NS）、邻居通告（NA）和重定向。

21. 设置M标记是告诉主机使用DHCPv6配置它的地址。0标记告诉主机使用DHCPv6去查找其他链路参数。

22. 可达时间字段指定了某个节点在确认邻居可达后应该假定它的邻居是可达的时间，以毫秒为单位。

23. 重传计时器字段指定了连续传送的邻居请求之间节点应该等待的时间，以毫秒为单位。

24. 在RA中路由器生存时间的值为0，表示始发路由器不应该增加到一台主机的缺省路由器列表中去。

25. 设置了S标记，表示NA是用来响应某个NS的。只有在NA用来响应一个请求时，才能表示双向可达已经确认，并且在邻居缓存中将邻居地址更改为可达状态；如果收到的是清除了S位的NA，则表示它是未被请求的，并不在邻居缓存中改变它的状态。

26. 全状态地址自动配置依靠DHCPv6给主机分配地址。无状态地址自动配置使用RA确定一个比链路本地更大范围的前缀，加上MAC-to-EUI64转换来确定主机地址。

27. MAC-to-EUI64转换在MAC地址中间插入一个值0xFFFE，接着反转U/L位为1，从而通过一个48位的MAC地址创建了一个64位接口ID。

28. 对于任意播地址从来不需要执行地址冲突检测。

29. 前缀FF02:0:0:0:0:1:FF00::/104用作被请求节点的多播地址。它加在被请求的地址的最后24位之前。

30. IPv6使用NDP的邻居地址解析功能代替了ARP，同时邻居缓存也代替了ARP缓存。

31. 私有地址是随机生成的接口ID，并在某些正常的周期或主机获取一个新的前缀时改变。它用来和一个自动配置的公共地址连在一起，确保主机的匿名性。公共地址用于可达性，而私有地址用于某台主机始发的所有数据包的源地址。

32. 不完全状态表示该条目的邻居地址解析正在处理。

33. Probe状态表示已经发送了一个NS去校验某个Stale状态的条目的双向可达性，但响应的NA还没有收到。

34. 邻居不可达性检测校验了一个邻居的双向可达性，要么通过来自收到发送消息的确认的上层协议的“提示（Hints）”；要么通过主动探测带有NS的邻居。

第3章

1. 路由表中的每一个表项至少要包括目标地址和下一跳的路由器地址，或者表明目标地址是直接相连的。
2. 这意味着路由器知道，对于相同的主IP地址的不同子网，有多个子网掩码。
3. 非连续子网指的是被一个不同的主IP地址分隔开的一个主IP网络地址的两个或多个子网。
4. **show ip route** 命令用来查看Cisco路由器的路由表。
5. 命令**show ipv6 route** 可以显示IPv6的路由表。显示的信息有前缀、前缀长度，以及下一跳地址或出站接口，同时还有管理距离和路由度量。
6. 第一个数字是学习到该路由的路由选择协议的管理距离，第二个数字是该路由的度量值。
7. 在静态路由表项中，如果使用本地接口来代替下一跳的地址，目标地址将作为直接连接的地址进入路由表。
8. 汇总路由是一个单独的路由表项，指向多个子网或主IP地址。对于静态路由，汇总路由能减少需要配置的静态路由表项。
9. 管理距离是一个路由选择协议或者静态路由的优先等级。每一个路由选择协议和静态路由都有管理距离值。当一台路由器从多个路由选择协议得知到达同一目标地址的多个路由选择表项，它将使用管理距离最小的那条路由。
10. 浮动静态路由是到达目标地址的备用路由。它的管理距离被设得很高，这样只有当别的优先级高的路由均不可用时，它才被派上用场。
11. 等价负载均衡把流量分布在具有相同度量值的多条路径上；非等价负载均衡把流量分布在具有不同度量值的多条路径上。流量将根据路由代价分配，代价高的分配得少，代价低的分配得多。

12. 如果在一个入站接口上配置了CEF，那么数据包将使用CEF进行交换，并使用CEF负载均衡规则：根据所做的配置，对于IPv4是每目的地或每数据包的，对于IPv6是每目的地的。如果CEF没有在入站接口上配置，那么出站接口将确定交换模式，以及负载均衡的方法。如果一个接口是快速交换，将执行每目标负载均衡；如果一个接口是处理交换，将执行每数据包负载均衡。

13. 当路由器需要转发数据包，但通过单一路由表查找却得不到它所需的所有信息，则会发生递归路由表查找。例如，路由器执行一次查找得到去往一个目标的路由，下一跳是路由器A，但它却不知道该如何到达路由器A，于是执行另外一次查找得到去往路由器A的路由。

第4章

1. 路由选择协议是路由器之间所讲的一种“语言”，用来共享网络目标地址的信息。
2. 一个路由选择协议至少要定义以下过程：
 - 将网络的可达信息传递给其他路由器；
 - 从其他路由器接收可达信息；
 - 根据它所拥有的可达信息决定最佳路由，在路由表中记录该信息；
 - 反应、补偿、宣告网络中的拓扑变化。
3. 路由的度量值，也叫做路由代价或者路由距离，用来决定到达一个目的地的最佳路径。最佳路由所使用的度量值类型定义。
4. 收敛时间：一组路由器所花费用来完成路由选择信息交换的时间。
5. 负载均衡是通过多条路径向同一目标地址传送数据包的过程。有4种类型的负载均衡：
 - 等价，每数据包；
 - 等价，每目的；
 - 非等价，每数据包；
 - 非等价，每目的。
6. 距离矢量协议：每台路由器依据它邻居的路由来计算路由，然后传递给其他的邻居。
7. 距离矢量协议存在的一些问题是：
 - 因为它依靠邻居得到正确路由选择信息，所以易产生不正确的信息；

- 收敛慢；
- 路由环路；
- 计数无限。

8. 邻居指的是连接至同一数据链路的路由器。

9. 当路由超出特定期限之后，路由无效计时器将它们从路由表中删除。

10. 使用简单的水平分隔，将不会发送路由信息到产生该路由信息的源点；具有毒性逆转的水平分隔虽然发送至源点，但把度量值设成不可达。

11. 当路由环路更新路由时会产生计数到天穷大的问题；每台路由器增加路由的度量值直到度量值达到无穷大。可以把“无穷大”定义为合理的较低的度量值，这样可以很快达到无穷大，路由也就宣布为不可达，由此可以控制计数到无穷大的影响。

12. 抑制计时器可用来预防路由选择环路。如果一条路由宣布为不可达或者度量值超过特定的阈值，路由器将停止接受有关该路由的其他信息，直到抑制计时器超时。这样，可以防止路由器在网络收敛时接受错误的路由选择信息。

13. 距离矢量路由器发送它整个的路由表，但它仅仅发送给直接连接的邻居。链路状态路由器仅仅发送有关它直接相连链路的信息，但它广播该信息至整个网络区域。距离矢量协议通常使用不同的Bellman—Ford算法来计算路由，链路状态协议通常使用不同的Dijkstra算法来计算路由。

14. 拓扑数据库保存了路由选择域中所有路由器产生的链路状态信息。

15. 每台路由器广播链路状态信息通告，该通告描述其自身的链路、自身链路的状态，以及整个网络区域中连接至这些链路的所有邻居。所有的路由器在链路状态数据库中保存所有收到的这些链路状态通告。每台路由器根据拓扑数据库中的信息计算最短路径树，并且基于此树往路由表中添加路由。

16. 序列号帮助路由器区分同一个链路状态通告的多个拷贝，并防止泛洪的链路状态通告在整个网络中无限循环。

17. Aging防止老的、可能过期的链路状态信息在拓扑数据库中停留时间过长，或被一台路由器接受。

18. 构建最短路径树时，路由器首先将它自己作为根，然后使用拓扑数据库中的信息，创建所有与它直连的邻居列表。到一个邻居的代价最小的路径将成为树的一个分枝，该路由器的所有邻居都被加入列表。检查该列表，看是否有重复路径；如果有，代价高的路径将从列表中删除，代价低的路由器将被加入树；该路由器的邻居也加入列表，再次检查该列表是否有重复路径。此过程不断重复，直到列表中没有路由器为止。

19. 在一个路由选择域中，区域是子域。由于限制了区域中每台路由器的链路状态数据库的大小，区域使得链路状态路由更加高效。

20. 根据使用时的需要，一个自主系统可被定义为一个在相同管理域中的网络，或者是一个单独的路由选择域。

21. 内部网关协议是在自主系统内部进行路由的路由选择协议；外部网关协议是在自主系统间进行路由的路由选择协议。

第5章

1. RIP使用UDP端口520。
2. RIP的度量值是跳数。不可达网络的跳数是16, RIP将它视为无限距离。
3. 每隔30s减去一个小的随机变量, RIP周期性地发送更新。这样可以防止和邻居的更新同步。
4. 如果6次更新都未收到, 则一个路由表项被标记为不可达。
5. 当一条路由宣布为不可达时, 启动垃圾收集计时器, 或者称为刷新计时器。当计时器超时, 该路由才从路由表中删除。这个过程使得一条不可达的路由在路由表中停留足够长的时间, 以便邻居都能够被通知到。
6. 随机计时器, 定时范围1~5s, 防止在拓扑改变时触发更新“风暴”。
7. 请求消息要求路由器进行更新。响应消息就是一个更新。
8. 请求消息可能要求一个完全的更新, 或者在一些特殊情况下, 它会要求特定的路由。
9. 当更新计时器超时, 或者当接受到一条请求消息时, 发出一条响应消息。
10. RIP更新不包括目标地址的子网掩码, 因此RIP路由器靠它自己接口的子网掩码来判断主网络地址如何被子网化。如果不配置特定的主网络地址, 路由器将不知道主网络是被子网化的; 因此, 不会将主网络地址的子网通告给其他的主网络。

第6章

1. 路由标记字段、子网掩码字段和下一跳字段是RIPv2的扩展，而在RIPv1消息中没有。RIP消息的基本格式并没有改变；版本2仅仅添加了一些域。
2. 除了使用一些新的域，RIPv2支持身份验证和组播更新。
3. RIPv2使用组播地址224.0.0.9。组播路由选择消息比广播好，因为主机和非RIPv2路由器将忽略组播消息。
4. 当另外的路由选择协议将RIPv2域作为它的传输域时，该协议便可以使用RIPv2的路由标记字段与RIPv2域另一端的对等体进行通信。
5. 下一跳字段用来把下一跳的地址通知给在相同多址网络上的其他路由器，该下一跳地址到达目标地址在度量值上要比原路由器近。
6. RIPv2和RIPv1使用相同的UDP端口，端口号520。
7. RIPv2使用UDP端口号为521。
8. 无类别路由选择协议不考虑在路由发现过程中的主网络地址，仅仅寻找最长匹配。
9. 为支持VLSM，路由选择协议在其更新中必须包括每个目标地址的子网掩码。
10. RIPv2的Cisco实现支持明文验证和MD5验证。RFC 2453仅仅定义了明文验证。

第7章

1. EIGRP是一种距离矢量协议。
2. 缺省时，EIGRP使用不超过50%的链路带宽，带宽配置在路由器的接口上。这个百分比可以通过命令**ip bandwidth-percent eigrp** 来改变。
3. EIGRP和IGRP使用相同的公式来计算复合度量值。但是EIGRP通过一个因子256 来扩展度量值。

EIGRP的4个基本部分是：

- 依赖于协议的模块；
 - 可靠传输协议；
 - 邻居发现和恢复模块；
 - 扩散更新算法。
4. 可靠传输意味着EIGRP数据包可以有保证、按次序地传输。RTP使用可靠组播，收到的数据包均被确认，以保证传输；使用序列号保证它们被有次序地传输。
 5. 序列号保证路由器接收的是最新的路由表项。
 6. EIGRP使用组播地址224.0.0.10。
 7. EIGRP使用的数据包类型是：
 - Hello数据包；
 - 确认数据包；
 - 更新数据包；
 - 查询数据包；

- 请求数据包;
- SIA-Queries;
- SIA-Replies。

8. 缺省的EIGRP Hello间隔是5s, 而在一些较慢的接口 (T-1或更低) 上, 这个时间是60s。

9. EIGRP的缺省抑制时间是Hello间隔的3倍。

10. 邻居表存储了有关EIGRP-speaking邻居的信息; 拓扑表列出了所有具有可行后继的已知路由。

11. 到一个目标网络的可行距离是, 路由器计算出的到达目标的最小距离。

12. 对一个目标网络来说, 可行后继是通过可行条件选出的。如果一个邻居通告的到达一个目标网络的距离比收到该通告的路由器到此目标网络的可行距离要小, 那么就满足可行条件。也就是说, 如果一台路由器的邻居到达目标网络的距离比该路由器要近, 那么这个邻居就是满足可行条件的。还有一种说法, 即相对于目标网络而言, 邻居是位于下游的。

13. 可行后继指的是一个满足可行条件的邻居。

14. 后继指的是目前正用来作为下一跳的可行后继。

15. 在特定路由器上, 如果它已经查询了它的邻居来寻找可行后继, 但是还没有收到任何一个邻居的回应时, 我们说, 这台路由器上的路由是活跃的; 当查询全部完成, 路由则是非活跃的。

16. 当拓扑表里没有可行后继时, 路由变为活跃的。

17. 从被查询的邻居收到一条回应时, 路由从活跃变为非活跃。

18. 当路由器在active time (缺省3min) 内没有收到被查询邻居的回应, 路由被宣布为stuck-in-active。收到一条代表邻居的具有无限量值的回应, 以满足DUAL, 然后该邻居从邻居表中删除。

19. 从一个IP网络地址产生一组子网地址，称为子网化。地址聚合指的是从一组网络或子网地址汇总出一个IP网络地址。

第8章

1. 从OSPF路由器的角度来说，邻居指的是直接与该OSPF路由器相邻的其他OSPF路由器。

2. OSPF邻接是指到一个邻居的概念上的链路，可以通过该链路传送LSA。

3. 有5种OSPF数据包类型，它们的作用如下：

- Hello，用来发现邻居，建立和保持邻接；
- Update，用来在邻居间发送LSA；
- 数据库描述数据包，被路由器用来在数据库同步的过程中向邻居描述它的链路状态数据库；
- 链路状态请求数据包，被路由器用来向邻居的链路状态数据库请求LSA。
- 链路状态确认数据包，用来保证可靠的LSA传输。

4. 路由器产生一个链路状态通告来描述一个或多个目标网络。OSPF的Update数据包将LSA从一个邻居传送至另外一个邻居。虽然LSA在整个OSPF域或区域（area）内广播，但Update数据包不会离开单个的数据链路。

5. 最常用的LSA类型和作用如下：

- 类型1（Router LSA）由每台路由器产生，用来描述产生它的路由器、该路由器的直连链路和状态，以及该路由器的邻居。
- 类型2（Network LSA）由多点访问链路上的指定路由器（Designated Router）产生，描述了该链路以及所有相连的邻居。
- 类型3（Network Summary LSA）由区域边界路由器（area border router）产生，描述了区域间的目标网络。

- 类型4（ASBR Summary LSA）由区域边界路由器产生，描述了区域外的自主系统边界路由器（autonomous system boundary router）。

- 类型5（AS External LSA）由自主系统边界路由器产生，描述OSPF域之外的目标网络。

- 类型7（NSSA External LSA）由非末梢区域内的自主系统边界路由器产生。

6. 路由器使用链路状态数据库来存储所有它知道的OSPF LSA，包括它自己的。数据库同步指的是保证一个区域内所有的路由器都有一致的链路状态数据库的过程。

7. 缺省的OSPF HelloInterval是10s。

8. 缺省的OSPF RouterDeadInterval是HelloInterval的4倍。

9. 路由器ID是一个OSPF路由器用来标识它自己的地址。它或者是路由器所有回环（loopback）接口的最高地址；或者如果没有配置loopback接口，就是路由器所有LAN接口的最高地址。它也可以手工配置。

10. 一个区域是一个OSPF子域，在区域内，所有的路由器都有一致的链路状态数据库。

11. 区域0是骨干区域。其他所有的区域必须通过主干区域来发送它们的区域内流量。

12. MaxAge, lh，是LSA被认为过期的时限。

13. 4种OSPF路由器类型是：

- 内部路由器（IR）：它所有的OSPF接口属于同一个区域；
- 骨干路由器（BR）：是区域0内的内部路由器；
- 区域边界路由器（ABR）：在多个区域内有OSPF接口；
- 自主系统边界路由器（ASBR）：宣告外部路由至OSPF域。

14. 4种OSPF路径类型是：

- 域内路径；
- 域间路径；
- 类型1外部路径；
- 类型2外部路径。

15. 5种OSPF网络类型为：

- 点到点网络；
- 广播型网络；
- 非广播型多址网络；
- 点到多点网络；
- 虚链路。

16. 指定路由器（DR）：代表了一个多路访问网络以及连接至这个网络和OSPF域的其余部分的路由器。

17. Cisco IOS这样计算一个接口的出站代价： $10^8 / \text{BW}$ ，其中BW是该接口上配置的带宽。 10^8 可以通过OSPF命令 **auto-cost reference-bandwidth** 改变。

18. 如果一个区域内的一台或者多台路由器不通过向区域外发送数据包就不能向区域内的其他路由器发送数据包，那么这个区域应该划分子区域。

19. 虚链路指的是通过一个非骨干区域扩展OSPF骨干连接的通道。

20. 末梢区域（stub area）是指类型5 LSA不能广播到的区域。完全末梢区域指的是类型3、类型4或类型5 LSA不能广播到的区域（除了用来通告缺省路由的类型3 LSA）。通过非末梢区域，外部的目标网络能够被通告至OSPF域内，但是类型5 LSA不能被ABR送至非末梢区域。

21. OSPF网络表项是路由表中的表项，描述了IP目标网络。OSPF路由器表项是在一个单独的路由表中的表项，该路由表只记录了到达ABR和ASBR的路由。

22. 类型2的认证使用MD5加密，类型1认证使用明文密码。

23. LSA报头中区别不同LSA的3个字段分别是类型、通告路由器、链路状态ID。LSA报头中区别相同LSA的不同实例的3个字段分别是序列号、老化时间及校验和。

第9章

1. 在编写本书的时候，OSPFv3还不能支持IPv4。为了能够在IPv4和IPv6之间进行路由选择，我们必须同时运行OSPFv2和OSPFv3。
2. 每条链路上多个实例意味着，在连接到同一条广播链路上的不同路由器之间可以形成各自不同的邻接；因此不同的OSPFv3路由选择域能够使用相同的共享链路，相互之间互不干扰。OSPFv3报头中的Instance ID字段实现了这个特性。
3. OSPFv3数据包可以使用内嵌的IPv6认证进行认证（依赖IPv6认证扩展报头）。OSPFv3不需要像OSPFv2那样有自己的认证机制。
4. OSPFv3下一报头号和OSPFv2的协议号相同，都是89。
5. OSPFv3使用保留的多播地址FF02::5（AllSPFRouters）和FF02::6（AllDRouters）。
6. 不是。OSPFv3使用和OSPFv2相同的5个消息数据包类型。
7. 第1位是U位，用来指出如果接收路由器收到未知类型的LSA应该如何处理。第2位和第3位是S位，用来表示LSA的泛洪扩散范围。
8. OSPFv3支持链路本地扩散范围，而OSPFv2不支持。链路LSA使用这个扩散范围。
9. OSPFv3路由器和网络LSA不通告前缀，而OSPFv2的路由器和网络LSA通告前缀。
10. 区域内前缀LSA携带的是与始发路由器相连的IPv6前缀。
11. 链路LSA携带的仅仅是两个直接连接的邻居之间的信息。

第10章

1. 中间系统是ISO称呼路由器的术语。
2. 网络协议数据单元是ISO称呼数据包的术语。
3. L1路由器与其他的区域没有直接连接。L2路由器与L1路由器之间没有邻接关系。L1/L2路由器路由区域间和区域内的流量，并且为L1路由器充当区域间网关。
4. Cisco路由器的缺省配置是L1/L2类型。
5. IS-IS区域的边界在路由器之间，在链路之上。OSPF区域的边界由路由器自己定义。
6. 两台具有相同AID的L1/L2路由器将同时形成L1和L2类型的邻接关系。两台具有不同AID的L1/L2路由器将只形成L2类型的邻接关系。
7. 两台L2类型的路由器将形成一个L2邻接关系，而AID可以相同或不同。
8. 网络实体标题是路由器标识自己和它所在区域的一个地址。
9. 在一个NET内NSAP选择符应该设成0x00。
10. 系统ID在IS-IS域内唯一地标识了一台路由器。
11. NET最后7个八位组字节前面的部分是区域地址。
12. IS-IS不选取BDR。
13. 伪节点ID是LAN ID的最后一个八位组字节。它的作用是为了区别由一台路由器产生的多个LAN ID，该路由器在多个LAN中是DR。
14. IS-IS LSP的MaxAge是1200s（20min）。MaxAgee（或开始的剩余生存时间）可以最大配置为65535s。

15. OSPF增加age至MaxAge;IS-IS减小age至0。一个新的OSPF LSA有age值0，而一个新的IS-IS LSP有age值MaxAge。
16. IS-IS路由器的刷新率是900s（15min）。
17. 一个完全序列号数据包（CSNP）列出了一个数据库中所有的LSP。在一个广播网络上，指定路由器周期性地发送CSNP来保持数据库的同步。
18. 一个部分序列号数据包列出了一个或多个LSP。有两个用途：在点到点网络上，它用来确认LSP的接收；在广播网络上，它用来请求LSP。
19. IS-IS路由器使用超载位通知它的邻居其内存过载了，并且不能存储整个链路状态数据库。
20. L1/L2路由器使用Attached位通知L1路由器它与L2骨干有连接。
21. Up/down位用来区分某个地址是区域内部始发的，还是泄漏到该区域的地址。
22. ISO指定4个度量值：缺省度量（Default）、开销度量（Expense）、时延度量（Delay）、差错度量（Error）。Cisco只支持Default。
23. 两种度量类型是普通度量值和扩展度量值，普通度量最大值是63，扩展度量的最大值为16 777 214。
24. 一个IS-IS路由的最大度量值，在普通度量类型下是1023，在扩展度量类型下是4 261 412 864。
25. L1 IS-IS度量值作用于区域内路由，L2 IS-IS度量值作用于区域间路由。
26. 内部度量值作用于目标网络在IS-IS域内的路由；外部度量值作用于目标网络在IS-IS域外的路由。
27. 即使在多拓扑模式下同时配置IPv4和IPv6，两台路由器之间也是形成单个邻接关系。

28. 在单台路由器上可以配置多个L1区域，只配置一个L2区域。

29. 两个活动的mesh组模式是Blocked和Set（Numbered）模式。
Blocked模式可以大大减少泛洪扩散，但可能减少冗余性，增加收敛时间。Set模式或编号mesh组减少的泛洪扩散负载不会像Blocked模式那么多，但是也降低了冗余性和收敛时间的潜在影响。

第11章

1. 从其他的路由选择协议、同一路由选择协议的两个进程之间、静态路由或者直连到目的网络学来的路由可以被重分配到一个路由选择域。路由也可以在IS-IS第1层和第2层之间重新分配。
2. 度量值用来在同一个路由选择协议产生的、到达同一目标网络的多条路由中间决定最佳路径；而管理距离用来在不同路由选择协议产生的、到达同一目标网络的多条路由中间决定最佳路径。
3. 在一个具有较高管理距离值的路由选择域中，一条路由可以被重新分配到具有较低管理距离值的路由选择域。如果该路由被分配回高管理距离域，数据包可能会错误地路由至低管理距离域。
4. 从无类别域到有类别域重新分配可变子网化的目标网络地址可能会有问题。有类别域可能不能识别来自无类别域重新分配的所有子网。
5. OSPF和IS-IS能理解缺省度量值，而RIP、IGRP和EIGRP则不能。
6. **metric** 命令为特定的重新分配语句引入了度量值。**default-metric** 语句为所有不包括**metric** 命令的重新分配语句引入度量值。
7. 如果没有**subnets** 关键字，只有不与路由器直连的主网络地址将被重新分配。
8. 产生汇总路由的路由器应该使用**null**接口作为汇总路由的下一跳。如果有一个数据包，只能匹配汇总路由，但不能匹配更加具体的、能够到达目的地址的路由，那么这个数据包将被丢弃。这防止了路由器转发丢失的数据包。

第12章

1. IPv4的缺省路由地址是0.0.0.0。
2. IPv6的缺省前缀/前缀长度是::/0。
3. EIGRP通告一个缺省地址作为外部地址类型。
4. 是的。
5. 末梢路由器是指只有一条链路连至其他路由器的路由器。末梢网络是指只连至一台路由器的网络。
6. 使用缺省路由（而不是完整的路由表）使得路由表很小，这样能够节省路由器内存，并且能限制必须被处理的路由选择信息，节省路由器处理周期。
7. 使用完整的路由表（而不是缺省路由）能使路由匹配更加精确。
8. ODR使用Cisco发现协议（CDP）来发现路由。
9. ODR在IOS 11.2和以后版本中可用。
10. ODR运行的介质必须支持SNAP。

第14章

1. 路由映射和访问列表相似，它们都定义了匹配的标准，以及匹配后采取的动作。它们的不同点在于路由映射不光定义了匹配标准，还定义了设置（set）标准。设置标准能修改一条路由或者根据数据包的参数路由一个数据包。
2. 策略路由是一种静态路由，它使用路由映射来决定哪些数据包应该转发，应该发往哪里。
3. 路由标记是路由选择信息数据包中的一些域，它们允许在路由选择域内部能够携带外部信息。
4. 路由标记不会影响携带它们的路由选择协议。

附录E

配置练习答案

第1章

1. 如果D类地址的前4位是1110，那么第一个八位组字节最小是11100000，最大是11101111。用十进制表示，分别为224和239。所以，D类地址的第一个八位组字节取值从224到239。

2. (a) 需要足够的子网位数 n ，使得 $2^n - 2 \geq 16000$ ；需要足够的主机位数 h ，使得 $2^h - 2 \geq 700$ 。子网掩码255.255.252.0为一个A类地址提供16382个子网，为其中每一个子网提供1022个主机地址。此掩码是惟一的答案。如果多一个子网位（255.255.254.0），将没有足够的主机地址。如果少一个子网位（255.255.248.0），将没有足够的子网。

(b) 需要足够的子网位数 n ，使得 $2^n - 2 \geq 500$ ；需要足够的主机位数 h ，使得 $2^h - 2 \geq 100$ 。子网掩码255.255.255.128为一个B类地址提供510个子网，为其中每一个子网提供126个主机地址。同样，此掩码是惟一的答案。

3. 用6位来子网化，一个C类地址将有 $2^6 - 2 = 62$ 个子网，每个子网有 $2^2 - 2 = 2$ 个主机地址。以这种模式，一个C类地址可以被62个点-to-点链路所使用。一个点到点链路只需要两个主机地址——链路的每一端一个。

4. 有28位掩码的C类地址可以有14个子网，每个子网有14个主机地址。首先给出子网。

这些子网是：

11111111111111111111111111110000 = 255.255.255.240（掩码）

110000001010100010010011**0001** 0000 = 192.168.147.16

110000001010100010010011**0010** 0000 = 192.168.147.32

110000001010100010010011**0011** 0000 = 192.168.147.48
110000001010100010010011**0100** 0000 = 192.168.147.64
110000001010100010010011**0101** 0000 = 192.168.147.80
110000001010100010010011**0110** 0000 = 192.168.147.96
110000001010100010010011**0111** 0000 = 192.168.147.112
110000001010100010010011**1000** 0000 = 192.168.147.128
110000001010100010010011**1001** 0000 = 192.168.147.144
110000001010100010010011**1010** 0000 = 192.168.147.160
110000001010100010010011**1011** 0000 = 192.168.147.176
110000001010100010010011**1100** 0000 = 192.168.147.192
110000001010100010010011**1101** 0000 = 192.168.147.208
110000001010100010010011**1110** 0000 = 192.168.147.224

下面给出每个子网的主机。每个子网的广播地址同样给出。

每个子网的主机地址是：

110000001010100010010011**0001** 0000 = 192.168.147.16（子网）
110000001010100010010011**0001** 0001 = 192.168.147.17
110000001010100010010011**0001** 0010 = 192.168.147.18
110000001010100010010011**0001** 0011 = 192.168.147.19
110000001010100010010011**0001** 0100 = 192.168.147.20
110000001010100010010011**0001** 0101 = 192.168.147.21

110000001010100010010011**0001** 0110 = 192.168.147.22
110000001010100010010011**0001** 0111 = 192.168.147.23
110000001010100010010011**0001** 1000 = 192.168.147.24
110000001010100010010011**0001** 1001 = 192.168.147.25
110000001010100010010011**0001** 1010 = 192.168.147.26
110000001010100010010011**0001** 1011 = 192.168.147.27
110000001010100010010011**0001** 1100 = 192.168.147.28
110000001010100010010011**0001** 1101 = 192.168.147.29
110000001010100010010011**0001** 1110 = 192.168.147.30
110000001010100010010011**0001** 1111 = 192.168.147.31 (广播)
110000001010100010010011**0010** 0000 = 192.168.147.32 (子网)
110000001010100010010011**0010** 0001 = 192.168.147.33
110000001010100010010011**0010** 0010 = 192.168.147.34
110000001010100010010011**0010** 0011 = 192.168.147.35
110000001010100010010011**0010** 0100 = 192.168.147.36
110000001010100010010011**0010** 0101 = 192.168.147.37
110000001010100010010011**0010** 0110 = 192.168.147.38
110000001010100010010011**0010** 0111 = 192.168.147.39
110000001010100010010011**0010** 1000 = 192.168.147.40
110000001010100010010011**0010** 1001 = 192.168.147.41

110000001010100010010011**0010** 1010 = 192.168.147.42
110000001010100010010011**0010** 1011 = 192.168.147.43
110000001010100010010011**0010** 1100 = 192.168.147.44
110000001010100010010011**0010** 1101 = 192.168.147.45
110000001010100010010011**0010** 1110 = 192.168.147.46
110000001010100010010011**0010** 1111 = 192.168.147.47 (广播)
110000001010100010010011**0011** 0000 = 192.168.147.48 (子网)
110000001010100010010011**0011** 0001 = 192.168.147.49
110000001010100010010011**0011** 0010 = 192.168.147.50
110000001010100010010011**0011** 0011 = 192.168.147.51
110000001010100010010011**0011** 0100 = 192.168.147.52
110000001010100010010011**0011** 0101 = 192.168.147.53
110000001010100010010011**0011** 0110 = 192.168.147.54
110000001010100010010011**0011** 0111 = 192.168.147.55
110000001010100010010011**0011** 1000 = 192.168.147.56
110000001010100010010011**0011** 1001 = 192.168.147.57
110000001010100010010011**0011** 1010 = 192.168.147.58
110000001010100010010011**0011** 1011 = 192.168.147.59
110000001010100010010011**0011** 1100 = 192.168.147.60
110000001010100010010011**0011** 1101 = 192.168.147.61

110000001010100010010011**0011** 1110 = 192.168.147.62

110000001010100010010011**0011** 1111 = 192.168.147.63 (广播)

110000001010100010010011**0100** 0000 = 192.168.147.64 (子网)

110000001010100010010011**0100** 0001 = 192.168.147.65

110000001010100010010011**0100** 0010 = 192.168.147.66

110000001010100010010011**0100** 0011 = 192.168.147.67

110000001010100010010011**0100** 0100 = 192.168.147.68

110000001010100010010011**0100** 0101 = 192.168.147.69

110000001010100010010011**0100** 0110 = 192.168.147.70

110000001010100010010011**0100** 0111 = 192.168.147.71

110000001010100010010011**0100** 1000 = 192.168.147.72

110000001010100010010011**0100** 1001 = 192.168.147.73

110000001010100010010011**0100** 1010 = 192.168.147.74

110000001010100010010011**0100** 1011 = 192.168.147.75

110000001010100010010011**0100** 1100 = 192.168.147.76

110000001010100010010011**0100** 1101 = 192.168.147.77

110000001010100010010011**0100** 1110 = 192.168.147.78

110000001010100010010011**0100** 1111 = 192.168.147.79 (广播)

110000001010100010010011**0101** 0000 = 192.168.147.80 (子网)

110000001010100010010011**0101** 0001 = 192.168.147.81

110000001010100010010011**0101** 0010 = 192.168.147.82
110000001010100010010011**0101** 0011 = 192.168.147.83
110000001010100010010011**0101** 0100 = 192.168.147.84
110000001010100010010011**0101** 0101 = 192.168.147.85
110000001010100010010011**0101** 0110 = 192.168.147.86
110000001010100010010011**0101** 0111 = 192.168.147.87
110000001010100010010011**0101** 1000 = 192.168.147.88
110000001010100010010011**0101** 1001 = 192.168.147.89
110000001010100010010011**0101** 1010 = 192.168.147.90
110000001010100010010011**0101** 1011 = 192.168.147.91
110000001010100010010011**0101** 1100 = 192.168.147.92
110000001010100010010011**0101** 1101 = 192.168.147.93
110000001010100010010011**0101** 1110 = 192.168.147.94
110000001010100010010011**0101** 1111 = 192.168.147.95 (广播)
110000001010100010010011**0110** 0000 = 192.168.147.96 (子网)
110000001010100010010011**0110** 0001 = 192.168.147.97
110000001010100010010011**0110** 0010 = 192.168.147.98
110000001010100010010011**0110** 0011 = 192.168.147.99
110000001010100010010011**0110** 0100 = 192.168.147.100
110000001010100010010011**0110** 0101 = 192.168.147.101

110000001010100010010011**0110** 0110 = 192.168.147.102
110000001010100010010011**0110** 0111 = 192.168.147.103
110000001010100010010011**0110** 1000 = 192.168.147.104
110000001010100010010011**0110** 1001 = 192.168.147.105
110000001010100010010011**0110** 1010 = 192.168.147.106
110000001010100010010011**0110** 1011 = 192.168.147.107
110000001010100010010011**0110** 1100 = 192.168.147.108
110000001010100010010011**0110** 1101 = 192.168.147.109
110000001010100010010011**0110** 1110 = 192.168.147.110
110000001010100010010011**0110** 1111 = 192.168.147.111 (广播)
110000001010100010010011**0111** 0000 = 192.168.147.112 (子网)
110000001010100010010011**0111** 0001 = 192.168.147.113
110000001010100010010011**0111** 0010 = 192.168.147.114
110000001010100010010011**0111** 0011 = 192.168.147.115
110000001010100010010011**0111** 0100 = 192.168.147.116
110000001010100010010011**0111** 0101 = 192.168.147.117
110000001010100010010011**0111** 0110 = 192.168.147.118
110000001010100010010011**0111** 0111 = 192.168.147.119
110000001010100010010011**0111** 1000 = 192.168.147.120
110000001010100010010011**0111** 1001 = 192.168.147.121

110000001010100010010011**0111** 1010 = 192.168.147.122
110000001010100010010011**0111** 1011 = 192.168.147.123
110000001010100010010011**0111** 1100 = 192.168.147.124
110000001010100010010011**0111** 1101 = 192.168.147.125
110000001010100010010011**0111** 1110 = 192.168.147.126
110000001010100010010011**0111** 1111 = 192.168.147.127 (广播)
110000001010100010010011**1000** 0000 = 192.168.147.128 (子网)
110000001010100010010011**1000** 0001 = 192.168.147.129
110000001010100010010011**1000** 0010 = 192.168.147.130
110000001010100010010011**1000** 0011 = 192.168.147.131
110000001010100010010011**1000** 0100 = 192.168.147.132
110000001010100010010011**1000** 0101 = 192.168.147.133
110000001010100010010011**1000** 0110 = 192.168.147.134
110000001010100010010011**1000** 0111 = 192.168.147.135
110000001010100010010011**1000** 1000 = 192.168.147.136
110000001010100010010011**1000** 1001 = 192.168.147.137
110000001010100010010011**1000** 1010 = 192.168.147.138
110000001010100010010011**1000** 1011 = 192.168.147.139
110000001010100010010011**1000** 1100 = 192.168.147.140
110000001010100010010011**1000** 1101 = 192.168.147.141

110000001010100010010011**1000** 1110 = 192.168.147.142

110000001010100010010011**1000** 1111 = 192.168.147.143 (广播)

110000001010100010010011**1001** 0000 = 192.168.147.144 (子网)

110000001010100010010011**1001** 0001 = 192.168.147.145

110000001010100010010011**1001** 0010 = 192.168.147.146

110000001010100010010011**1001** 0011 = 192.168.147.147

110000001010100010010011**1001** 0100 = 192.168.147.148

110000001010100010010011**1001** 0101 = 192.168.147.149

110000001010100010010011**1001** 0110 = 192.168.147.150

110000001010100010010011**1001** 0111 = 192.168.147.151

110000001010100010010011**1001** 1000 = 192.168.147.152

110000001010100010010011**1001** 1001 = 192.168.147.153

110000001010100010010011**1001** 1010 = 192.168.147.154

110000001010100010010011**1001** 1011 = 192.168.147.155

110000001010100010010011**1001** 1100 = 192.168.147.156

110000001010100010010011**1001** 1101 = 192.168.147.157

110000001010100010010011**1001** 1110 = 192.168.147.158

110000001010100010010011**1001** 1111 = 192.168.147.159 (广播)

110000001010100010010011**1010** 0000 = 192.168.147.160 (子网)

110000001010100010010011**1010** 0001 = 192.168.147.161

110000001010100010010011**1010** 0010 = 192.168.147.162
110000001010100010010011**1010** 0011 = 192.168.147.163
110000001010100010010011**1010** 0100 = 192.168.147.164
110000001010100010010011**1010** 0101 = 192.168.147.165
110000001010100010010011**1010** 0110 = 192.168.147.166
110000001010100010010011**1010** 0111 = 192.168.147.167
110000001010100010010011**1010** 1000 = 192.168.147.168
110000001010100010010011**1010** 1001 = 192.168.147.169
110000001010100010010011**1010** 1010 = 192.168.147.170
110000001010100010010011**1010** 1011 = 192.168.147.171
110000001010100010010011**1010** 1100 = 192.168.147.172
110000001010100010010011**1010** 1101 = 192.168.147.173
110000001010100010010011**1010** 1110 = 192.168.147.174
110000001010100010010011**1010** 1111 = 192.168.147.175 (广播)
110000001010100010010011**1011** 0000 = 192.168.147.176 (子网)
110000001010100010010011**1011** 0001 = 192.168.147.177
110000001010100010010011**1011** 0010 = 192.168.147.178
110000001010100010010011**1011** 0011 = 192.168.147.179
110000001010100010010011**1011** 0100 = 192.168.147.180
110000001010100010010011**1011** 0101 = 192.168.147.181

110000001010100010010011**1011** 0110 = 192.168.147.182
110000001010100010010011**1011** 0111 = 192.168.147.183
110000001010100010010011**1011** 1000 = 192.168.147.184
110000001010100010010011**1011** 1001 = 192.168.147.185
110000001010100010010011**1011** 1010 = 192.168.147.186
110000001010100010010011**1011** 1011 = 192.168.147.187
110000001010100010010011**1011** 1100 = 192.168.147.188
110000001010100010010011**1011** 1101 = 192.168.147.189
110000001010100010010011**1011** 1110 = 192.168.147.190
110000001010100010010011**1011** 1111 = 192.168.147.191 (广播)
110000001010100010010011**1100** 0000 = 192.168.147.192 (子网)
110000001010100010010011**1100** 0001 = 192.168.147.193
110000001010100010010011**1100** 0010 = 192.168.147.194
110000001010100010010011**1100** 0011 = 192.168.147.195
110000001010100010010011**1100** 0100 = 192.168.147.196
110000001010100010010011**1100** 0101 = 192.168.147.197
110000001010100010010011**1100** 0110 = 192.168.147.198
110000001010100010010011**1100** 0111 = 192.168.147.199
110000001010100010010011**1100** 1000 = 192.168.147.200
110000001010100010010011**1100** 1001 = 192.168.147.201

110000001010100010010011**1100** 1010 = 192.168.147.202
110000001010100010010011**1100** 1011 = 192.168.147.203
110000001010100010010011**1100** 1100 = 192.168.147.204
110000001010100010010011**1100** 1101 = 192.168.147.205
110000001010100010010011**1100** 1110 = 192.168.147.206
110000001010100010010011**1100** 1111 = 192.168.147.207 (广播)
110000001010100010010011**1101** 0000 = 192.168.147.208 (子网)
110000001010100010010011**1101** 0001 = 192.168.147.209
110000001010100010010011**1101** 0010 = 192.168.147.210
110000001010100010010011**1101** 0011 = 192.168.147.211
110000001010100010010011**1101** 0100 = 192.168.147.212
110000001010100010010011**1101** 0101 = 192.168.147.213
110000001010100010010011**1101** 0110 = 192.168.147.214
110000001010100010010011**1101** 0111 = 192.168.147.215
110000001010100010010011**1101** 1000 = 192.168.147.216
110000001010100010010011**1101** 1001 = 192.168.147.217
110000001010100010010011**1101** 1010 = 192.168.147.218
110000001010100010010011**1101** 1011 = 192.168.147.219
110000001010100010010011**1101** 1100 = 192.168.147.220
110000001010100010010011**1101** 1101 = 192.168.147.221

110000001010100010010011**1101** 1110 = 192.168.147.222

110000001010100010010011**1101** 1111 = 192.168.147.223 (广播)

110000001010100010010011**1110** 0000 = 192.168.147.224 (子网)

110000001010100010010011**1110** 0001 = 192.168.147.225

110000001010100010010011**1110** 0010 = 192.168.147.226

110000001010100010010011**1110** 0011 = 192.168.147.227

110000001010100010010011**1110** 0100 = 192.168.147.228

110000001010100010010011**1110** 0101 = 192.168.147.229

110000001010100010010011**1110** 0110 = 192.168.147.230

110000001010100010010011**1110** 0111 = 192.168.147.231

110000001010100010010011**1110** 1000 = 192.168.147.232

110000001010100010010011**1110** 1001 = 192.168.147.233

110000001010100010010011**1110** 1010 = 192.168.147.234

110000001010100010010011**1110** 1011 = 192.168.147.235

110000001010100010010011**1110** 1100 = 192.168.147.236

110000001010100010010011**1110** 1101 = 192.168.147.237

110000001010100010010011**1110** 1110 = 192.168.147.238

110000001010100010010011**1110** 1111 = 192.168.147.239 (广播)

5. 这里给出此题的答案，可以看出比上一题短，因为没有写出每一个子网的每一台主机地址。

有29位掩码的C类地址意味着将有30个子网，每个子网有6个主机地址。

这些子网是:

11111111111111111111111111111111**1111** 000 = 255.255.255.248 (mask)

110000001010100010010011**00001** 000 = 192.168.147.8

110000001010100010010011**00010** 000 = 192.168.147.16

110000001010100010010011**00011** 000 = 192.168.147.24

110000001010100010010011**00100** 000 = 192.168.147.32

110000001010100010010011**00101** 000 = 192.168.147.40

110000001010100010010011**00110** 000 = 192.168.147.48

110000001010100010010011**00111** 000 = 192.168.147.56

110000001010100010010011**01000** 000 = 192.168.147.64

110000001010100010010011**01001** 000 = 192.168.147.72

110000001010100010010011**01010** 000 = 192.168.147.80

110000001010100010010011**01011** 000 = 192.168.147.88

110000001010100010010011**01100** 000 = 192.168.147.96

110000001010100010010011**01101** 000 = 192.168.147.104

110000001010100010010011**01100** 000 = 192.168.147.112

110000001010100010010011**01111** 000 = 192.168.147.120

110000001010100010010011**10000** 000 = 192.168.147.128

110000001010100010010011**10001** 000 = 192.168.147.136

110000001010100010010011**10010** 000 = 192.168.147.144

110000001010100010010011**10011** 000 = 192.168.147.152

110000001010100010010011**10100** 000 = 192.168.147.160

110000001010100010010011**10101** 000 = 192.168.147.168

110000001010100010010011**10110** 000 = 192.168.147.176

110000001010100010010011**10111** 000 = 192.168.147.184

110000001010100010010011**11000** 000 = 192.168.147.192

110000001010100010010011**11001** 000 = 192.168.147.200

110000001010100010010011**11010** 000 = 192.168.147.208

110000001010100010010011**11011** 000 = 192.168.147.216

110000001010100010010011**11100** 000 = 192.168.147.224

110000001010100010010011**11101** 000 = 192.168.147.232

110000001010100010010011**11110** 000 = 192.168.147.240

下面给出每个子网的广播地址（将每个子网的主机位全置为1）。

子网广播地址是：

110000001010100010010011**100001** 111 = 192.168.147.15

110000001010100010010011**100010** 111 = 192.168.147.23

110000001010100010010011**100011** 111 = 192.168.147.31

110000001010100010010011**100100** 111 = 192.168.147.39

110000001010100010010011**100101** 111 = 192.168.147.47

110000001010100010010011**100110** 111 = 192.168.147.55

110000001010100010010011**00111** 111 = 192.168.147.63
110000001010100010010011**01000** 111 = 192.168.147.71
110000001010100010010011**01001** 111 = 192.168.147.79
110000001010100010010011**01010** 111 = 192.168.147.87
110000001010100010010011**01011** 111 = 192.168.147.95
110000001010100010010011**01100** 111 = 192.168.147.103
110000001010100010010011**01101** 111 = 192.168.147.111
110000001010100010010011**01100** 111 = 192.168.147.119
110000001010100010010011**01111** 111 = 192.168.147.127
110000001010100010010011**10000** 111 = 192.168.147.135
110000001010100010010011**10001** 111 = 192.168.147.143
110000001010100010010011**10010** 111 = 192.168.147.151
110000001010100010010011**10011** 111 = 192.168.147.159
110000001010100010010011**10100** 111 = 192.168.147.167
110000001010100010010011**10101** 111 = 192.168.147.175
110000001010100010010011**10110** 111 = 192.168.147.183
110000001010100010010011**10111** 111 = 192.168.147.191
110000001010100010010011**11000** 111 = 192.168.147.199
110000001010100010010011**11001** 111 = 192.168.147.207
110000001010100010010011**11010** 111 = 192.168.147.215

110000001010100010010011**11011** 111 = 192.168.147.223

110000001010100010010011**11100** 111 = 192.168.147.231

110000001010100010010011**11101** 111 = 192.168.147.239

110000001010100010010011**11110** 111 = 192.168.147.247

最后，给出每个子网的主机地址，它们是介于子网地址和子网广播地址之间的地址。

子 网	广 播	主 机 地 址
192.168.147.8	192.168.147.15	192.168.147.9 - 192.168.147.14
192.168.147.16	192.168.147.23	192.168.147.17 - 192.168.147.22
192.168.147.24	192.168.147.31	192.168.147.25 - 192.168.147.30
192.168.147.32	192.168.147.39	192.168.147.33 - 192.168.147.38
192.168.147.40	192.168.147.47	192.168.147.41 - 192.168.147.46
192.168.147.48	192.168.147.55	192.168.147.49 - 192.168.147.54
192.168.147.56	192.168.147.63	192.168.147.57 - 192.168.147.62
192.168.147.64	192.168.147.71	192.168.147.65 - 192.168.147.70
192.168.147.72	192.168.147.79	192.168.147.73 - 192.168.147.78
192.168.147.80	192.168.147.87	192.168.147.81 - 192.168.147.86
192.168.147.88	192.168.147.95	192.168.147.89 - 192.168.147.94
192.168.147.96	192.168.147.103	192.168.147.97 - 192.168.147.102
192.168.147.104	192.168.147.111	192.168.147.105 - 192.168.147.110
192.168.147.112	192.168.147.119	192.168.147.113 - 192.168.147.118
192.168.147.120	192.168.147.127	192.168.147.121 - 192.168.147.126
192.168.147.128	192.168.147.135	192.168.147.129 - 192.168.147.134
192.168.147.136	192.168.147.143	192.168.147.137 - 192.168.147.142
192.168.147.144	192.168.147.151	192.168.147.145 - 192.168.147.150
192.168.147.152	192.168.147.159	192.168.147.153 - 192.168.147.158
192.168.147.160	192.168.147.167	192.168.147.161 - 192.168.147.166
192.168.147.168	192.168.147.175	192.168.147.169 - 192.168.147.174
192.168.147.176	192.168.147.183	192.168.147.177 - 192.168.147.182

192.168.147.184 192.168.147.191 192.168.147.185 - 192.168.147.190
192.168.147.192 192.168.147.199 192.168.147.193 - 192.168.147.198
192.168.147.200 192.168.147.207 192.168.147.201 - 192.168.147.206
192.168.147.208 192.168.147.215 192.168.147.209 - 192.168.147.214
192.168.147.216 192.168.147.223 192.168.147.217 - 192.168.147.222
192.168.147.224 192.168.147.231 192.168.147.225 - 192.168.147.230
192.168.147.232 192.168.147.239 192.168.147.233 - 192.168.147.238
192.168.147.240 192.168.147.247 192.168.147.241 - 192.168.147.246

6. 一个有20位掩码的B类地址将有14个子网，每个子网有4 094个主机地址。这些子网是：

11111111111111111111**1111** 000000000000 = 255.255.240.0（掩码）

1010110000010000**0001** 000000000000 = 172.16.16.0

1010110000010000**0010** 000000000000 = 172.16.32.0

1010110000010000**0011** 000000000000 = 172.16.48.0

1010110000010000**0100** 000000000000 = 172.16.64.0

1010110000010000**0101** 000000000000 = 172.16.80.0

1010110000010000**0110** 000000000000 = 172.16.96.0

1010110000010000**0111** 000000000000 = 172.16.112.0

1010110000010000**1000** 000000000000 = 172.16.128.0

1010110000010000**1001** 000000000000 = 172.16.144.0

1010110000010000**1010** 000000000000 = 172.16.160.0

1010110000010000**1011** 000000000000 = 172.16.176.0

1010110000010000**1100** 000000000000 = 172.16.192.0

1010110000010000**1101** 000000000000 = 172.16.208.0

1010110000010000**1110** 000000000000 = 172.16.224.0

子网广播地址是：

1010110000010000**0001** 111111111111 = 172.16.31.255

1010110000010000**0010** 111111111111 = 172.16.47.255

1010110000010000**0011** 111111111111 = 172.16.63.255

1010110000010000**0100** 111111111111 = 172.16.79.255

1010110000010000**0101** 111111111111 = 172.16.95.255

1010110000010000**0110** 111111111111 = 172.16.111.255

1010110000010000**0111** 111111111111 = 172.16.127.255

1010110000010000**1000** 111111111111 = 172.16.143.255

1010110000010000**1001** 111111111111 = 172.16.159.255

1010110000010000**1010** 111111111111 = 172.16.175.255

1010110000010000**1011** 111111111111 = 172.16.191.255

1010110000010000**1100** 111111111111 = 172.16.207.255

1010110000010000**1101** 111111111111 = 172.16.223.255

1010110000010000**1110** 111111111111 = 172.16.239.255

使用以上的子网和广播地址，主机地址是：

子 网	广 播	主 机 地 址
172.16.16.0	172.16.31.255	172.16.16.1-172.16.31.254
172.16.32.0	172.16.47.255	172.16.32.1-172.16.47.254

172.16.48.0	172.16.63.255	172.16.48.1-172.16.63.254
172.16.64.0	172.16.79.255	172.16.64.1-172.16.79.254
172.16.80.0	172.16.95.255	172.16.80.1-172.16.95.254
172.16.96.0	172.16.111.255	172.16.96.1-172.16.111.254
172.16.112.0	172.16.127.255	172.16.112.1-172.16.127.254
172.16.128.0	172.16.143.255	172.16.128.1-172.16.143.254
172.16.144.0	172.16.159.255	172.16.144.1-172.16.159.254
172.16.160.0	172.16.175.255	172.16.160.1-172.16.175.254
172.16.176.0	172.16.191.255	172.16.176.1-172.16.191.254
172.16.192.0	172.16.207.255	172.16.192.1-172.16.207.254
172.16.208.0	172.16.223.255	172.16.208.1-172.16.223.254
172.16.224.0	172.16.239.255	172.224.16.1-172.16.239.254

第3章

1. 首先确定每个链路的子网地址，然后写出静态路由。记住路由器的路由表中将缺省包含直接相连的子网。静态路由是：

RTA

```
ip route 192.168.2.64 255.255.255.224 192.168.2.131  
ip route 192.168.2.160 255.255.255.224 192.168.2.131  
ip route 192.168.1.128 255.255.255.240 192.168.2.131  
ip route 192.168.1.16 255.255.255.240 192.168.2.131  
ip route 192.168.2.32 255.255.255.224 192.168.2.131  
ip route 192.168.1.160 255.255.255.240 192.168.2.131  
ip route 10.1.1.0 255.255.255.0 192.168.2.131  
ip route 10.1.3.0 255.255.255.0 192.168.2.131  
ip route 10.1.2.0 255.255.255.0 192.168.2.131
```

RTB

```
ip route 10.1.4.0 255.255.255.0 192.168.2.132  
ip route 192.168.1.128 255.255.255.240 192.168.2.174  
ip route 192.168.1.16 255.255.255.240 192.168.2.174  
ip route 192.168.2.32 255.255.255.224 192.168.2.174  
ip route 192.168.1.160 255.255.255.240 192.168.2.174  
ip route 10.1.1.0 255.255.255.0 192.168.2.174
```

ip route 10.1.3.0 255.255.255.0 192.168.2.174

ip route 10.1.2.0 255.255.255.0 192.168.2.174

RTC

ip route 10.1.4.0 255.255.255.0 192.168.2.185

ip route 192.168.2.128 255.255.255.224 192.168.2.185

ip route 192.168.2.64 255.255.255.224 192.168.2.185

ip route 192.168.2.32 255.255.255.224 192.168.1.20

ip route 10.1.1.0 255.255.255.0 192.168.1.173

ip route 10.1.3.0 255.255.255.0 192.168.1.173

ip route 10.1.2.0 255.255.255.0 192.168.1.173

RTD

ip route 10.1.4.0 255.255.255.0 192.168.1.29

ip route 192.168.2.128 255.255.255.224 192.168.1.29

ip route 192.168.2.64 255.255.255.224 192.168.1.29

ip route 192.168.2.160 255.255.255.224 192.168.1.29

ip route 192.168.1.128 255.255.255.240 192.168.1.29

ip route 192.168.1.160 255.255.255.240 192.168.1.29

ip route 10.1.1.0 255.255.255.0 192.168.1.29

ip route 10.1.3.0 255.255.255.0 192.168.1.29

ip route 10.1.2.0 255.255.255.0 192.168.1.29

RTE

```
ip route 10.1.4.0 255.255.255.0 192.168.1.163
ip route 192.168.2.128 255.255.255.224 192.168.1.163
ip route 192.168.2.64 255.255.255.224 192.168.1.163
ip route 192.168.2.160 255.255.255.224 192.168.1.163
ip route 192.168.1.128 255.255.255.240 192.168.1.163
ip route 192.168.1.16 255.255.255.240 192.168.1.163
ip route 192.168.2.32 255.255.255.224 192.168.1.163
ip route 10.1.2.0 255.255.255.0 10.1.3.2
```

RTF

```
ip route 10.1.4.0 255.255.255.0 10.1.3.1
ip route 192.168.2.128 255.255.255.224 10.1.3.1
ip route 192.168.2.64 255.255.255.224 10.1.3.1
ip route 192.168.2.160 255.255.255.224 10.1.3.1
ip route 192.168.1.128 255.255.255.240 10.1.3.1
ip route 192.168.1.16 255.255.255.240 10.1.3.1
ip route 192.168.2.32 255.255.255.224 10.1.3.1
ip route 192.168.1.160 255.255.255.240 10.1.3.1
ip route 10.1.1.0 255.255.255.0 10.1.3.1
```

2. 静态路由是：

RTA

ip route 192.168.0.0 255.255.0.0 192.168.2.131

ip route 10.1.0.0 255.255.0.0 192.168.2.131

RTB

ip route 10.1.4.0 255.255.255.0 192.168.2.132

ip route 192.168.0.0 255.255.0.0 192.168.2.174

ip route 10.1.0.0 255.255.0.0 192.168.2.174

RTC

ip route 10.1.4.0 255.255.255.0 192.168.2.185

ip route 192.168.2.32 255.255.255.224 192.168.1.20

ip route 10.1.0.0 255.255.0.0 192.168.1.173

ip route 192.168.2.64 255.255.255.224 192.168.2.185

ip route 192.168.2.128 255.255.255.224 192.168.2.185

RTD

ip route 10.1.0.0 255.255.0.0 192.168.1.29

ip route 192.168.0.0 255.255.0.0 192.168.1.29

RTE

ip route 10.1.4.0 255.255.255.0 192.168.1.163

ip route 192.168.0.0 255.255.0.0 192.168.1.163

ip route 10.1.2.0 255.255.255.0 10.1.3.2

RTF

ip route 10.1.0.0 255.255.0.0 10.1.3.1

ip route 192.168.0.0 255.255.0.0 10.1.3.1

3. 静态路由是：

RTA

ip route 172.16.7.0 255.255.255.0 172.16.2.2

ip route 172.16.7.0 255.255.255.0 172.16.4.2 50

ip route 172.16.6.0 255.255.255.0 172.16.2.2

ip route 172.16.6.0 255.255.255.0 172.16.4.2 50

ip route 172.16.8.0 255.255.255.0 172.16.4.2

ip route 172.16.8.0 255.255.255.0 172.16.2.2 50

ip route 172.16.5.0 255.255.255.0 172.16.4.2

ip route 172.16.5.0 255.255.255.0 172.16.2.2 50

ip route 172.16.9.0 255.255.255.0 172.16.2.2

ip route 172.16.9.0 255.255.255.0 172.16.4.2

RTB

ip route 172.16.1.0 255.255.255.0 172.16.2.1

ip route 172.16.1.0 255.255.255.0 172.16.6.1 50

ip route 172.16.4.0 255.255.255.0 172.16.2.1

ip route 172.16.4.0 255.255.255.0 172.16.6.1 50

```
ip route 172.16.9.0 255.255.255.0 172.16.6.1  
ip route 172.16.9.0 255.255.255.0 172.16.2.1 50  
ip route 172.16.5.0 255.255.255.0 172.16.6.1  
ip route 172.16.5.0 255.255.255.0 172.16.2.1 50  
ip route 172.16.8.0 255.255.255.0 172.16.6.1  
ip route 172.16.8.0 255.255.255.0 172.16.2.1
```

RTC

```
ip route 172.16.1.0 255.255.255.0 172.16.6.2  
ip route 172.16.1.0 255.255.255.0 172.16.5.1  
ip route 172.16.4.0 255.255.255.0 172.16.5.1  
ip route 172.16.4.0 255.255.255.0 172.16.6.2 50  
ip route 172.16.2.0 255.255.255.0 172.16.6.2  
ip route 172.16.2.0 255.255.255.0 172.16.5.1 50  
ip route 172.16.7.0 255.255.255.0 172.16.6.2  
ip route 172.16.7.0 255.255.255.0 172.16.5.1 50  
ip route 172.16.8.0 255.255.255.0 172.16.5.1  
ip route 172.16.8.0 255.255.255.0 172.16.6.2 50
```

RTD

```
ip route 172.16.1.0 255.255.255.0 172.16.4.1  
ip route 172.16.1.0 255.255.255.0 172.16.5.2 50
```

ip route 172.16.2.0 255.255.255.0 172.16.4.1

ip route 172.16.2.0 255.255.255.0 172.16.5.2 50

ip route 172.16.9.0 255.255.255.0 172.16.5.2

ip route 172.16.9.0 255.255.255.0 172.16.4.1 50

ip route 172.16.6.0 255.255.255.0 172.16.5.2

ip route 172.16.6.0 255.255.255.0 172.16.4.1 50

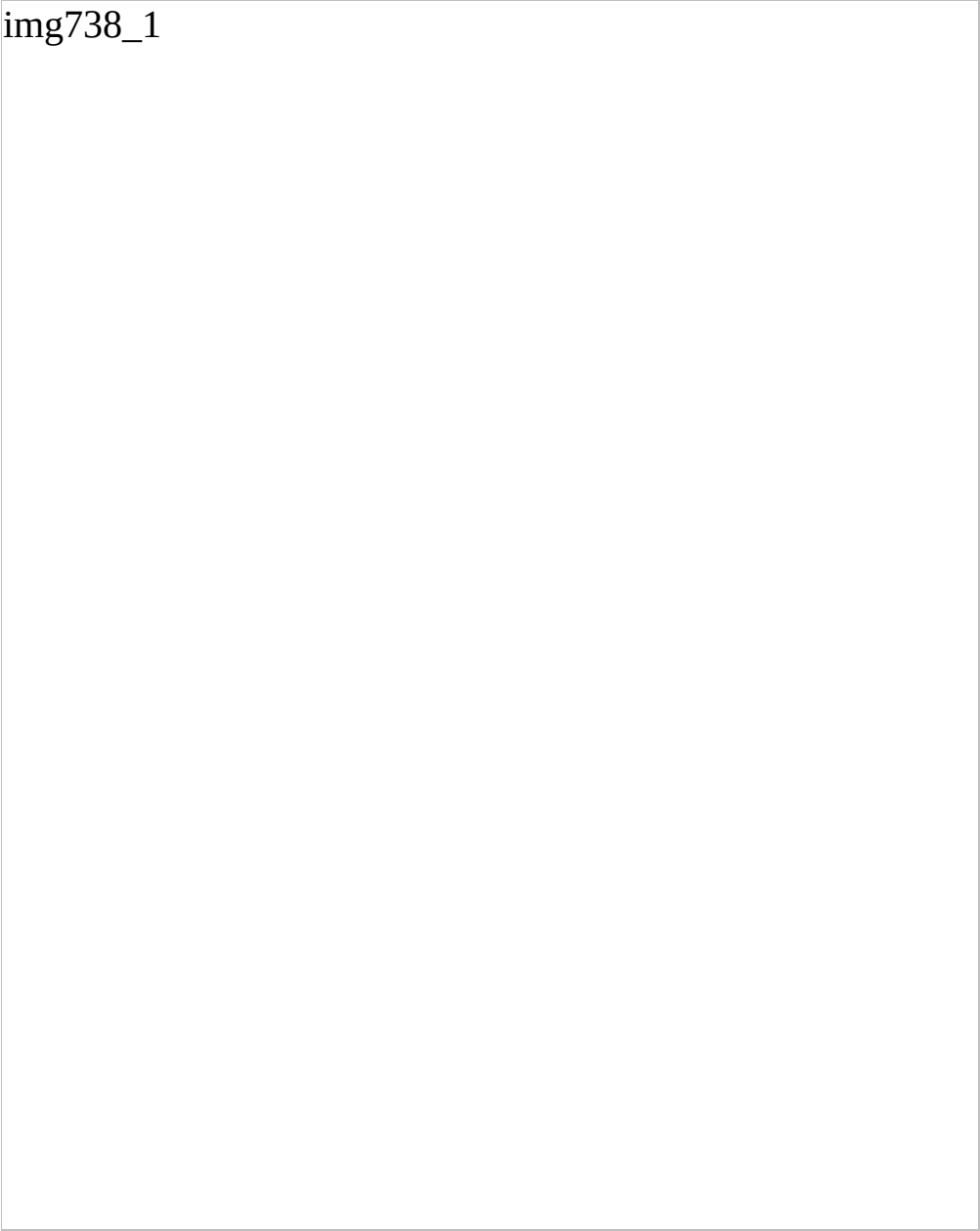
ip route 172.16.7.0 255.255.255.0 172.16.5.2

ip route 172.16.7.0 255.255.255.0 172.16.4.1

第5章

1. 除了这里给出的RIP配置，在RTE和RTF之间还必须使用2级地址配置192.168.5.0的一个子网。否则子网192.168.5.192/27和192.168.5.96/27不连续。RIP配置如下：

img738_1



img739_1

2. 为了在RTC和RTD之间单播RIP更新，配置为：

img739_2

3. 更新时间作用于整个RIP进程。如果串行链路的更新时间变了，路由器其他链路的更新时间也要改变。这就意味着邻居路由器的计时器要发生变化，邻居路由器的邻居的计时器也要顺次变化，等等。在一台路由器上改变更新计时器的级联效应导致RIP域中每台路由器的计时器都要变化。在每台路由器上增加RIP更新周期至2min的命令为：

timers basic 120 360 360 480

因为更新计时器改变了，所以无效计时器、抑制计时器和刷新计时器也必须改变。像缺省的那样，把无效计时器和抑制计时器设为更新计时器的6倍，将使得这个网络的转换时间很长。所以，把无效计时器和抑制计时器设为更新计时器的3倍。刷新计时器必须比抑制计时器长，所以

这里把它设成长60s。

4. 从RTA到网络192.168.4.0有两跳，所以给度量值增加14将会使路由的度量值达到16（不可达）。记得在配置练习1中，192.168.5.0的一个子网必须在和192.168.4.0相同的链路上使用2级地址来配置，这样192.168.5.0的子网才连续。因此，从RTB到192.168.5.0也是两跳。假设RTA和RTB上连接到RTC的接口都是E0，配置为：

RTA

router rip

offset-list 1 in 14 Ethernet0

network 192.168.2.0

!

access-list 1 permit 192.168.4.0 0.0.0.0

RTB

router rip

offset-list 1 in 14 Ethernet0

network 192.168.2.0

!

access-list 1 permit 192.168.5.0 0.0.0.0

5. RTB有更长一点的掩码，能正确解释所有子网。问题在RTA上。RTA有27位掩码，它把RTB的子网192.168.20.40/29和192.168.20.49/29解释为192.168.20.32/27——与它的直连链路相同的子网。因此，RTA没有整个网络的正确视图。

但是，如果代理ARP启用了，数据包仍然能被路由。例如，假设RTA要路由一个目的地址是192.168.20.50的数据包。RTA错误地把这个地址当

作是它的子网192.168.20.32/27的成员，并且在该子网上发出ARP请求，要求得到192.168.20.50的MAC地址。RTB听到ARP请求，它正确地解释这个地址为它的子网192.168.20.48/29的一个成员，并且回应它在192.168.20.32/29上的接口的MAC地址。RTA然后向RTB转发数据包，RTB再将这个数据包转发至正确目的地。如果代理ARP没有启用，数据包将不会被正确地从RTA转发至RTB。

第6章

1. Taos的RIP配置中可增加**neighbor 172.25.150.206** 语句，使Taos单播RIP更新至该地址。该措施仅当下面情况满足时有效。Pojoaque的RIPv1进程符合这样的规则：版本号高于1的RIP消息的未用域被忽略，剩下的数据包被处理。

2. 首先计算有最大主机数的子网。然后利用未用的子网位，计算有次最大主机数的子网，等等。记住，当用位组来表示子网时，没有连续的子网能以相同的位组开始。例如，如果第一个子网开始于00，所有后续的子网必须开始于01、10或11。如果第二个子网开始于010，后续的子网不能开始于010。

一种解答如下（子网位用黑体表示）：

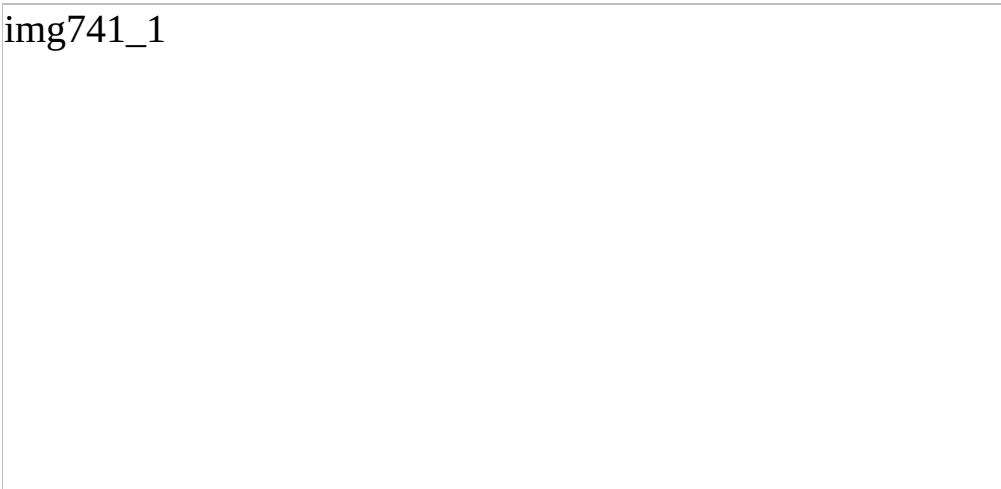
00000000	192.168.100.0/26（62 hosts）
01000000	192.168.100.64/27（30 hosts）
01100000	192.168.100.96/28（14 hosts）
01110000	192.168.100.112/28（14 hosts）
10000000	192.168.100.128/28（14 hosts）
10010000	192.168.100.144/28（14 hosts）
10100000	192.168.100.160/28（14 hosts）
10110000	192.168.100.176/29（8 hosts）
10111000	192.168.100.184/29（8 hosts）
11000000	192.168.100.192/29（8 hosts）
11001000	192.168.100.200/29（8 hosts）
11010000	192.168.100.208/29（8 hosts）
11011000	192.168.100.216/30（2 hosts）
11011100	192.168.100.220/30（2 hosts）
11100000	192.168.100.224/30（2 hosts）
11100100	192.168.100.228/30（2 hosts）
11101000	192.168.100.232/30（2 hosts）
11101100	192.168.100.236/30（2 hosts）

11110000	192.168.100.240/30 (2 hosts)
11110100	192.168.100.244/30 (2 hosts)
11111000	192.168.100.248/30 (2 hosts)
11111100	192.168.100.252/30 (2 hosts)

3. RTA、RTB和RTD的RIP配置中有**version 2** 语句。另外，RTA和RTB的RIP配置中包含**no auto-summary** 语句。RTA和RTB上连接到子网192.168.2.64/28的接口配置了语句**ip rip send version 1 2** 和**ip rip receive version 1 2**。RTD上连接到子网192.168.2.128/28的接口配置了语句**ip rip send version 1** 和**ip rip receive version 1**。

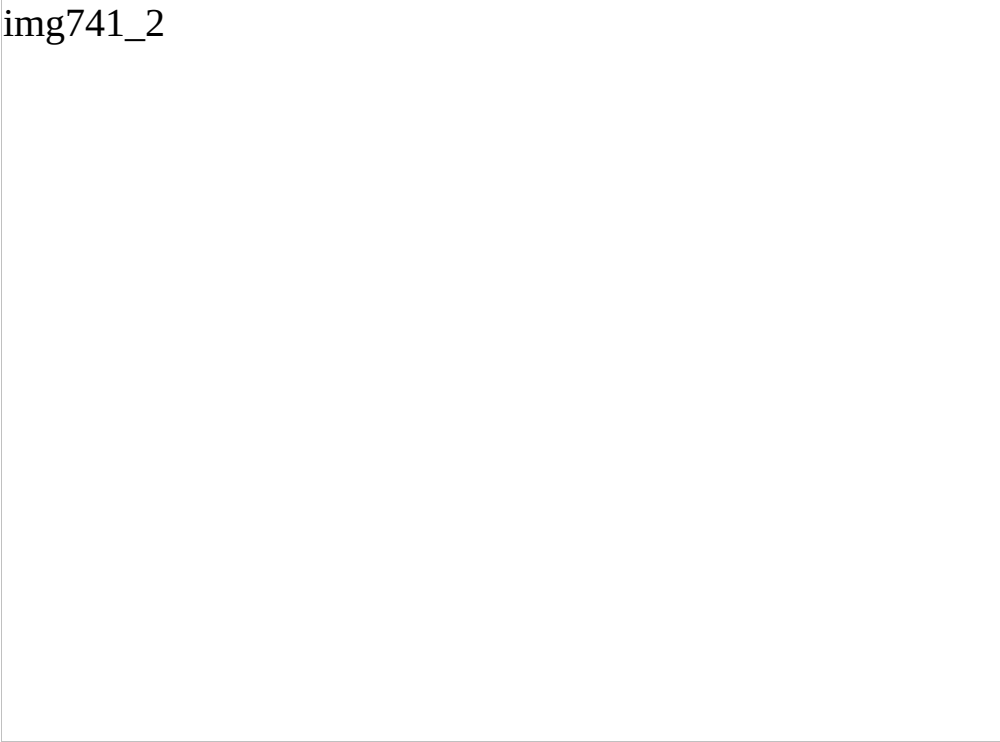
4. 下面的解答在RTB和RTD上使用一个密钥链*CCIE* 以及一个密钥串*exercise4*。假设两台路由器的接口都是SO，RTB和RTD的配置如下：

img741_1



5. 这里的解答假设配置练习4中的验证密钥在2004年10月31日午夜启动生效。这里使用的第二个密钥串是*exercise5a*， 第三个是*exercise5b*。

img741_2



6. 全局配置**ipv6 unicast-routing**。在LAN端口上配置IPv6地址。在路由器RTA和RTB共享的LAN上配置地址2001:DB8:0:2::/64。在所有的IPv6地址接口上配置RIPng路由选择进程。

第7章

1. EIGRP配置如下:

RTA

```
router eigrp 5
network 172.16.0.0
network 172.18.0.0
```

RTB

```
router eigrp 5
network 172.16.0.0
```

RTC

```
router eigrp 5
network 172.16.0.0
network 172.17.0.0
```

2. 下面的解答使用了一个密钥链*CCIE* 以及密钥串*exercise2a* 和 *exercise2b*。假设今天的日期是2004年11月30日，第一个密钥8:30AM开始启用，串口的配置为:

```
key chain CCIE
key 1
key-string exercise2a
accept-lifetime 08:30:00 Dec 2 2004 08:30:00 Jan 1 2005
```


send-lifetime 08:30:00 Dec 2 2004 08:30:00 Jan 1 2005

key 2

key-string exercise2b

accept-lifetime 08:30:00 Jan 1 2005 infinite

send-lifetime 08:30:00 Jan 1 2005 infinite

!

interface Serial0

ip address 172.16.3.19X 255.255.255.252

ip authentication key-chain eigrp 5 CCIE

ip authentication mode eigrp 5 md5

3. RTD的EIGRP配置为:

router eigrp 5

network 172.16.0.0

network 172.17.0.0

no auto-summary

在RTC上自动汇总必须被关掉。

4. RTF上的EIGRP配置为:

router eigrp 5

network 172.16.0.0

network 172.18.0.0

no auto-summary

RTA在子网172.18.10.96/27上的接口上加入语句**ip summary-address eigrp 5172.18.10.192 255.255.255.224**。而且，注意到自动汇总在RTA上必须关掉，因为RTF上出现了网络172.16.0.0。

5. RTA能送给RTF 172.16.3.128/25的汇总地址，能送给RTB 172.16.3.0/25的汇总地址。所有其他的汇总都是自动执行。

第8章

1. OSPF配置为:

RTA

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
```

RTB

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.5.0.0 0.0.255.255 area 5
area 5 virtual-link 10.100.100.9
```

RTC

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.10.0.0 0.0.255.255 area 10
network 10.30.0.0 0.0.255.255 area 30
```

RTD

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.20.0.0 0.0.255.255 area 20
```

RTE

router ospf 1

network 10.0.0.0 0.0.255.255 area 0

network 10.15.0.0 0.0.255.255 area 15

RTF

router ospf 1

network 10.5.0.0 0.0.255.255 area 5

RTG

router ospf 1

network 10.10.1.58 0.0.0.0 area 10

RTH

router ospf 1

network 10.20.100.100 0.0.0.0 area 20

network 10.20.1.0 0.0.0.255 area 20

RTI

router ospf 1

network 10.5.0.0 0.0.255.255 area 5

network 10.35.0.0 0.0.255.255 area 35

area 5 virtual-link 10.100.100.2

RTJ

```
router ospf 1
```

```
network 10.15.0.0 0.0.255.255 area 15
```

RTK到RTN有帧中继接口。到帧中继网络的4个接口都在同一个子网上，因此OSPF网络类型必须是广播或是点到多点的：

RTK

```
interface Serial0
```

```
encapsulation frame-relay
```

```
ip address 10.30.254.193 255.255.255.192
```

```
ip ospf network point-to-multipoint
```

```
!
```

```
router ospf 1
```

```
network 10.30.0.0 0.0.255.255 area 30
```

RTL

```
encapsulation frame-relay
```

```
ip address 10.30.254.194 255.255.255.192
```

```
ip ospf network point-to-multipoint
```

```
!
```

```
router ospf 1
```

```
network 10.30.0.0 0.0.255.255 area 30
```

RTM

```
encapsulation frame-relay
```

ip address 10.30.254.195 255.255.255.192

ip ospf network point-to-multipoint

!

router ospf 1

network 10.30.0.0 0.0.255.255 area 30

RTN

encapsulation frame-relay

ip address 10.30.254.196 255.255.255.192

ip ospf network point-to-multipoint

!

router ospf 1

network 10.30.0.0 0.0.255.255 area 30

2. ABR配置为:

RTB

router ospf 1

network 10.0.0.0 0.0.255.255 area 0

network 10.5.0.0 0.0.255.255 area 5

area 5 virtual-link 10.100.100.9

area 0 range 10.0.0.0 255.255.0.0

area 5 range 10.5.0.0 255.255.0.0

RTC

router ospf 1

network 10.0.0.0 0.0.255.255 area 0

network 10.10.0.0 0.0.255.255 area 10

network 10.30.0.0 0.0.255.255 area 30

area 0 range 10.0.0.0 255.255.0.0

area 10 range 10.10.0.0 255.255.0.0

area 30 range 10.30.0.0 255.255.0.0

RTD

router ospf 1

network 10.0.0.0 0.0.255.255 area 0

network 10.20.0.0 0.0.255.255 area 20

area 0 range 10.0.0.0 255.255.0.0

area 20 range 10.20.0.0 255.255.0.0

RTE

router ospf 1

network 10.0.0.0 0.0.255.255 area 0

network 10.15.0.0 0.0.255.255 area 15

area 0 range 10.0.0.0 255.255.0.0

area 15 range 10.15.0.0 255.255.0.0

RTI

router ospf 1

network 10.5.0.0 0.0.255.255 area 5

network 10.35.0.0 0.0.255.255 area 35

area 5 virtual-link 10.100.100.2

area 0 range 10.0.0.0 255.255.0.0

area 5 range 10.5.0.0 255.255.0.0

area 35 range 10.35.0.0 255.255.0.0

3. 配置为:

RTE

router ospf 1

network 10.0.0.0 0.0.255.255 area 0

network 10.15.0.0 0.0.255.255 area 15

area 15 stub

RTJ

router ospf 1

network 10.15.0.0 0.0.255.255 area 15

area 15 stub

4. 配置为:

RTC


```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.10.0.0 0.0.255.255 area 10
network 10.30.0.0 0.0.255.255 area 30
area 0 range 10.0.0.0 255.255.0.0
area 10 range 10.10.0.0 255.255.0.0
area 30 stub no-summary
area 30 range 10.30.0.0 255.255.0.0
```

RTK

```
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
area 30 stub
```

RTL

```
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
area 30 stub
```

RTM

```
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
area 30 stub
```

RTN

```
router ospf 1
```

```
network 10.30.0.0 0.0.255.255 area 30
```

```
area 30 stub
```

5. 配置为:

RTD

```
router ospf 1
```

```
network 10.0.0.0 0.0.255.255 area 0
```

```
network 10.20.0.0 0.0.255.255 area 20
```

```
area 20 nssa
```

RTH

```
router ospf 1
```

```
network 10.20.100.100 0.0.0.0 area 20
```

```
area 20 nssa
```

6. 点到点线路只有一个端点需要被配置为按需电路。在本题解答中，RTC被选择:

```
interface Serial0
```

```
ip address 10.30.255.249 255.255.255.252
```

```
ip ospf demand-circuit
```

```
!
```

```
router ospf 1
```

network 10.0.0.0 0.0.255.255 area 0

network 10.10.0.0 0.0.255.255 area 10

network 10.30.0.0 0.0.255.255 area 30

area 0 range 10.0.0.0 255.255.0.0

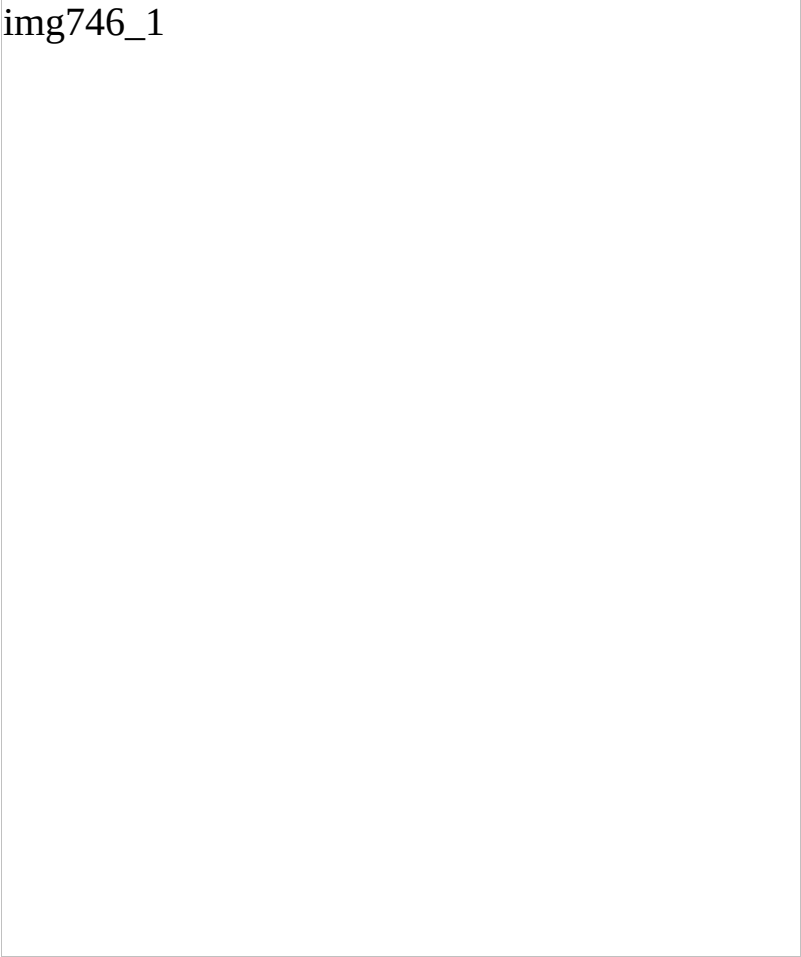
area 10 range 10.10.0.0 255.255.0.0

area 30 range 10.30.0.0 255.255.0.0

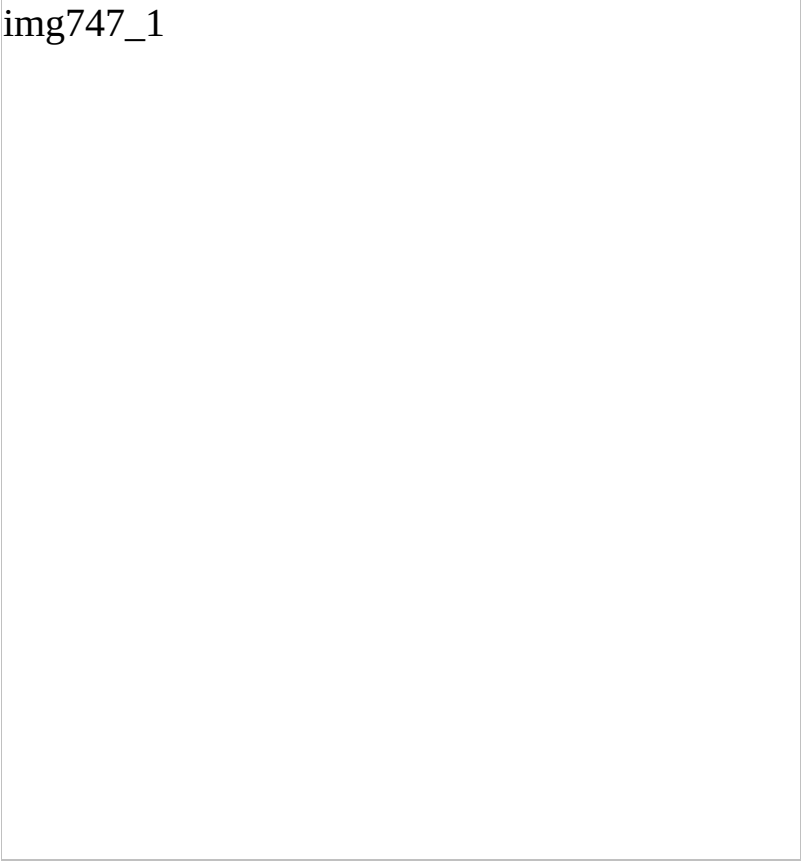
第9章

1. 对于已经配置到某个运行OSPFv3进程的接口上的辅助IPv6前缀地址，不需要再配置另外的OSPFv3命令。也就是说，如果在某个接口上配置了OSPFv3，在这个接口上增加一个辅助IPv6地址，那么这个辅助地址就会被自动地加入到OSPFv3进程中。
2. 是的，两台路由器如果配置了不同的前缀也能够形成邻接关系。在Hello数据包中不包含前缀信息，Hello数据包的源地址是链路本地地址。配置在接口上的其他IPv6地址和这两台路由器是否形成邻接关系无关。
3. 路由器的基本配置如下：

img746_1




img747_1



4. 以下新的路由器配置将区域1变为一个完全末梢区域，并汇总到区域0:

img747_2



第10章

1. 本题答案是使用系统ID 0000.1234.abcdX, X使得路由器在IS-IS域中惟一。系统ID有6个八位组字节, 是Cisco IOS所要求的。最小的NET长度是8个八位组字节, 其中一个是SEL, 所以区域地址是一个八位组字节的数字, 对应于表10.6中的区域号。配置为:

RTA

```
interface Ethernet0

ip address 192.168.1.17 255.255.255.240

ip router isis

!

interface Ethernet1

ip address 192.168.1.50 255.255.255.240

ip router isis

!

router isis

net 00.0000.1234.abcd.00

is-type level-1
```

RTB

```
interface Ethernet0

ip address 192.168.1.33 255.255.255.240

ip router isis
```

!

interface Ethernet1

ip address 192.168.1.51 255.255.255.240

ip router isis

!

router isis

net 00.0000.1234.abc2.00

is-type level-1

RTC

interface Ethernet0

ip address 192.168.1.49 255.255.255.240

ip router isis

!

interface Serial0

ip address 192.168.1.133 255.255.255.252

ip router isis

!

router isis

net 00.0000.1234.abc3.00

RTD

interface Serial0

ip address 192.168.1.134 255.255.255.252

ip router isis

!

interface Serial1

ip address 192.168.1.137 255.255.255.252

ip router isis

!

router isis

net 02.0000.1234.abc4.00

is-type level-2-only

RTE

interface Serial0

ip address 192.168.1.142 255.255.255.252

ip router isis

!

interface Serial1

ip address 192.168.1.145 255.255.255.252

ip router isis

!

interface Serial2

ip address 192.168.1.138 255.255.255.252

ip router isis

!

router isis

net 02.0000.1234.abc5.00

is-type level-2-only

RTF

interface Serial0

ip address 192.168.1.141 255.255.255.252

ip router isis

!

interface Serial1

ip address 192.168.1.158 255.255.255.252

ip router isis

!

router isis

net 02.0000.1234.abc6.00

is-type level-2-only

RTG

interface Ethernet0

ip address 192.168.1.111 255.255.255.224

ip router isis

!

interface Serial0

ip address 192.168.1.157 255.255.255.252

ip router isis

!

router isis

net 01.0000.1234.abc7.00

RTH

interface Ethernet0

ip address 192.168.1.73 255.255.255.224

ip router isis

!

interface Ethernet1

ip address 192.168.1.97 255.255.255.224

ip router isis

!

router isis

net 01.0000.1234.abc8.00

is-type level-1

RTI

interface Ethernet0

ip address 192.168.1.225 255.255.255.248

ip router isis

!

interface Ethernet1

ip address 192.168.1.221 255.255.255.248

ip router isis

!

interface Serial0

ip address 192.168.1.249 255.255.255.252

ip router isis

!

interface Serial1

ip address 192.168.1.146 255.255.255.252

ip router isis

!

router isis

net 03.0000.1234.abc9.00

RTJ

interface Ethernet0

ip address 192.168.1.201 255.255.255.248

ip router isis

!

interface Ethernet1

ip address 192.168.1.217 255.255.255.248

ip router isis

!

router isis

net 03.0000.1234.abca.00

is-type level-1

RTK

interface Ethernet0

ip address 192.168.1.209 255.255.255.248

ip router isis

!

interface Serial0

ip address 192.168.1.250 255.255.255.252

```
ip router isis  
  
!  
  
router isis  
  
net 03.0000.1234.abcb.00  
  
is-type level-1
```

2. 配置为：

RTA

```
ipv6 unicast-routing  
  
interface Ethernet0  
  
ip address 192.168.1.17 255.255.255.240  
  
ip router isis  
  
ipv6 address 2001:db8:0:10::1/64  
  
ipv6 router isis  
  
!  
  
interface Ethernet1  
  
ip address 192.168.1.50 255.255.255.240  
  
ip router isis  
  
ipv6 address 2001:db8:0:30::2/64  
  
ipv6 router isis  
  
!
```

router isis

net 00.0000.1234.abcd.00

is-type level-1

metric-style wide

address-family ipv6

multi-topology

RTB

ipv6 unicast-routing

interface Ethernet0

ip address 192.168.1.33 255.255.255.240

ip router isis

ipv6 address 2001:db8:0:20::1/64

ipv6 router isis

!

interface Ethernet1

ip address 192.168.1.51 255.255.255.240

ip router isis

ipv6 address 2001:db8:0:30::3/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc2.00

is-type level-1

metric-style wide

address-family ipv6

multi-topology

RTC

ipv6 unicast-routing

interface Ethernet0

ip address 192.168.1.49 255.255.255.240

ip router isis

ipv6 address 2001:db8:0:30::1/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.133 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:84::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc3.00

metric-style wide

address-family ipv6

multi-topology

RTD

ipv6 unicast-routing

interface Serial0

ip address 192.168.1.134 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:84::2/64

ipv6 router isis

!

interface Serial1

ip address 192.168.1.137 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:88::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc4.00

is-type level-2-only

metric-style wide

address-family ipv6

multi-topology

RTE

ipv6 unicast-routing

interface Serial0

ip address 192.168.1.142 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:8c::2/64

ipv6 router isis

!

interface Serial1

ip address 192.168.1.145 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:90::1/64

ipv6 router isis

!

interface Serial2

ip address 192.168.1.138 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:88::2/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc5.00

is-type level-2-only

metric-style wide

address-family ipv6

multi-topology

RTF

ipv6 unicast-routing

interface Serial0

ip address 192.168.1.141 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:8c::1/64

ipv6 router isis

!

interface Serial1

ip address 192.168.1.158 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:9c::2/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc6.00

is-type level-2-only

metric-style wide

address-family ipv6

multi-topology

RTG

ipv6 unicast-routing

interface Ethernet0

ip address 192.168.1.111 255.255.255.224

ip router isis

ip router clns

ipv6 address 2001:db8:0:60::f/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.157 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:9c::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc7.00

metric-style wide

address-family ipv6

multi-topology

RTH

ipv6 unicast-routing

interface Ethernet0

ip address 192.168.1.73 255.255.255.224

ip router isis

ipv6 address 2001:db8:0:40::9/64

ipv6 router isis

!

interface Ethernet1

ip address 192.168.1.97 255.255.255.224

ip router isis

ip router clns

ipv6 address 2001:db8:0:60::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc8.00

is-type level-1

metric-style wide

address-family ipv6

multi-topology

RTI

ipv6 unicast-routing

interface Ethernet0

ip address 192.168.1.225 255.255.255.248

ip router isis

ipv6 address 2001:db8:0:00::1/64

ipv6 router isis

!

interface Ethernetl

ip address 192.168.1.221 255.255.255.248

ip router isis

ipv6 address 2001:db8:0:d8::5/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.249 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:f8::1/64

ipv6 router isis

!

interface Serial1

ip address 192.168.1.146 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:90::2/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc9.00

metric-style wide

address-family ipv6

multi-topology

RTJ

ipv6 unicast-routing

interface Ethernet0

ip address 192.168.1.201 255.255.255.248

ip router isis

ipv6 address 2001:db8:0:c8::1/64

ipv6 router isis

!

interface Ethernet1

ip address 192.168.1.217 255.255.255.248

ip router isis

ipv6 address 2001:db8:0:d8::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abca.00

is-type level-1

metric-style wide

address-family ipv6

multi-topology

RTK

ipv6 unicast-routing

interface Ethernet0

ip address 192.168.1.209 255.255.255.248

ip router isis

ipv6 address 2001:db8:0:d0::1/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.250 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:f8::2/64

ipv6 router isis

!

router isis

net 00.0000.1234.abcb.00

is-type level-1

metric-style wide

address-family ipv6

multi-topology

3. 配置为:

RTD

Ipv6 unicast-routing

Key chain Fair

Key 1

Key-string Eiffel

interface Serial0

ip address 192.168.1.134 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:84::2/64

ipv6 router isis

!

interface Serial1

ip address 192.168.1.137 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:88::1/64

ipv6 router isis

authentication key-chain Fair level-2

authentication mode md5 level-2

!

router isis

net 00.0000.1234.abc4.00

is-type level-2-only

metric-style wide

address-family ipv6

multi-topology

RTE

Ipv6 unicast-routing

Key chain Fair

Key 1

Key-string Eiffel

Key chain Paris

Key 1

Key-string Tower

interface Serial0

ip address 192.168.1.142 255.255.255.252

ip router isis

ipv6 address 2001:db7:0:8c::2/64

ipv6 router isis

authentication key-chain Paris level-2

authentication mode md5 level-2

!

interface Serial1

ip address 192.168.1.145 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:90::1/64

ipv6 router isis

!

interface Serial2

ip address 192.168.1.138 255.255.255.252

ip router isis ipv6 address 2001:db8:0:88::2/64

ipv6 router isis

authentication key-chain Fair level-2

authentication mode md5 level-2

!

router isis

net 00.0000.1234.abc5.00

is-type level-2-only

metric-style wide

address-family ipv6

multi-topology

RTF

Ipv6 unicast-routing

Key chain Paris

Key 1

Key-string Tower

interface Serial0

ip address 192.168.1.141 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:8c::1/64

ipv6 router isis

authentication key-chain Paris level-2

authentication mode md5 level-2

!

interface Serial1

ip address 192.168.1.158 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:9c::2/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc6.00

is-type level-2-only

metric-style wide

address-family ipv6

multi-topology

4. 配置为:

RTG

Ipv6 unicast-routing

interface Ethernet0

ip address 192.168.1.111 255.255.255.224

ip router isis

ipv6 address 2001:db8:0:96::f/64

ipv6 router isis

ip router clns

!

interface Serial0

ip address 192.168.1.157 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:9c::1/64

ipv6 router isis!

router isis

net 00.0000.1234.abc7.00

area-password Scotland

metric-style wide

address-family ipv6

multi-topology

RTH

Ipv6 unicast-routing

interface Ethernet0

ip address 192.168.1.73 255.255.255.224

ip router isis

ipv6 address 2001:db8:0:40::9/64

ipv6 router isis

!

interface Ethernet1

ip address 192.168.1.97 255.255.255.224

ip router isis

ip router clns

ipv6 address 2001:db8:0:60::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc8.00

is-type level-1

area-password Scotland

metric-style wide

address-family ipv6

multi-topology

5. 配置为:

RTC

Ipv6 unicast-routing

Key chain Austria

Key 1

Key-string Vienna

interface Ethernet0

ip address 192.168.1.49 255.255.255.240

ip router isis

ipv6 address 2001:db8:0:30::1/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.133 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:84::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc3.00

authentication key-chain Austria level-2

authentication mode text

metric-style wide

address-family ipv6

multi-topology

RTD

Ipv6 unicast-routing

Key chain Fair

Key 1

Key-string Eiffel

Key chain Austria

Key 1

Key-string Vienna

interface Serial0

ip address 192.168.1.134 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:84::2/64

ipv6 router isis

!

interface Serial1

ip address 192.168.1.137 255.255.255.252

ip router isis

authentication key-chain Fair level-2

authentication mode md5 level-2

ipv6 address 2001:db8:0:88::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc4.00

is-type level-2-only

authentication key-chain Austria level-2

authentication mode text

metric-style wide

address-family ipv6

multi-topology

RTE

Ipv6 unicast-routing

Key chain Fair

Key 1

Key-string Eiffel

Key chain Paris

Key 1

Key-string Tower

Key chain Austria

Key 1

Key-string Vienna

interface Serial0

ip address 192.168.1.142 255.255.255.252

ip router isis

authentication key-chain Paris level-2

authentication mode md5 level-2

ipv6 address 2001:db8:0:8c::2

ipv6 routing isis

!

interface Serial1

ip address 192.168.1.145 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:90::1/64

ipv6 router isis

!

interface Serial2

ip address 192.168.1.138 255.255.255.252

ip router isis

authentication key-chain Fair level-2

authentication mode md5 level-2

ipv6 address 2001:db8:0:88::2/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc5.00

is-type level-2-only

authentication key-chain Austria level-2

authentication mode text

metric-style wide

address-family ipv6

multi-topology

RTF

Ipv6 unicast-routing

Key chain Paris

Key 1

Key-string Tower

Key chain Austria

Key 1

Key-string Vienna

interface Serial0

ip address 192.168.1.141 255.255.255.252

ip router isis

authentication key-chain Paris level-2

authentication mode md5 level-2

ipv6 address 2001:db8:0:8c::1/64

ipv6 router isis

!

interface Serial1

ip address 192.168.1.158 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:9c::2/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc6.00

is-type level-2-only

authentication key-chain Austria level-2

authentication mode text

metric-style wide

address-family ipv6

multi-topology

RTG

Ipv6 unicast-routing

Key chain Austria

Key 1

Key-string Vienna

interface Ethernet0

ip address 192.168.1.111 255.255.255.224

ip router isis

ip router clns

ipv6 address 2001:db8:0:96::f/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.157 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:9c::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc7.00

area-password Scotland

authentication key-chain Austria level-2

authentication mode text

metric-style wide

address-family ipv6

multi-topology

RTI

Ipv6 unicast-routing

Key chain Austria

Key 1

Key-string Vienna

interface Ethernet0

ip address 192.168.1.225 255.255.255.248

ip router isis

ipv6 address 2001:db8:0:e0::1/64

ipv6 router isis

!

interface Ethernet1

ip address 192.168.1.221 255.255.255.248

ip router isis

ipv6 address 2001:db8:0:d8::5/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.249 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:f8::1/64

ipv6 router isis

!

interface Serial1

ip address 192.168.1.146 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:90::2/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc9.00

authentication key-chain Austria level-2

authentication mode text

metric-style wide

address-family ipv6

multi-topology

6. 配置为:

RTC

Ipv6 unicast-routing

Key chain Austria

Key 1

Key-string Vienna

interface Ethernet0

ip address 192.168.1.49 255.255.255.240

ip router isis

ipv6 address 2001:db8:0:30::1/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.133 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:84::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc3.00

summary-address 192.168.1.0 255.255.255.192

authentication key-chain Austria level-2

authentication mode text

metric-style wide

address-family ipv6

multi-topology

summary-prefix 2001:db8::/58

RTG

Ipv6 unicast-routing

Key chain Austria

Key 1

Key-string Vienna

interface Ethernet0

ip address 192.168.1.111 255.255.255.224

ip router isis

ip router clns

ipv6 address 2001:db8:0:96::f/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.157 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:9c::1/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc7.00

area-password Scotland

authentication key-chain Austria level-2

authentication mode text

metric-style wide

summary-address 192.168.1.64 255.255.255.192

address-family ipv6

multi-topology

summary-prefix 2001:db8:0:40::/58

RTI

Ipv6 unicast-routing

Key chain Austria

Key 1

Key-string Vienna

interface Ethernet0

ip address 192.168.1.225 255.255.255.248

ip router isis

ipv6 address 2001:db8:0:e0::1/64

ipv6 router isis

!

interface Ethernet1

ip address 192.168.1.221 255.255.255.248

ip router isis

ipv6 address 2001:db8:0:d8::5/64

ipv6 router isis

!

interface Serial0

ip address 192.168.1.249 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:f8::1/64

ipv6 router isis

!

interface Serial1

ip address 192.168.1.146 255.255.255.252

ip router isis

ipv6 address 2001:db8:0:90::2/64

ipv6 router isis

!

router isis

net 00.0000.1234.abc9.00

authentication key-chain Austria level-2

authentication mode text

metric-style wide

summary-address 192.168.1.192 255.255.255.192

address-family ipv6

multi-topology

summary-prefix 2001:db8:0:c0::/58

第11章

1. RIP不宣告子网172.16.1.144/28。为了纠正这个问题，对这个子网加一条带27位掩码的静态路由。因为此掩码涉及RTB的E0接口，所以该路由被自动分配至RIP中。

```
interface Ethernet0
```

```
ip address 172.16.1.146 255.255.255.240
```

```
ipv6 address 2001:db8:0:190::2/64
```

```
ipv6 ospf 1 area 0
```

```
!
```

```
interface Ethernet1
```

```
ip address 172.16.1.98 255.255.255.224
```

```
ipv6 address 2001:db8:0:160::2/64
```

```
ipv6 rip ripdomain enable
```

```
!
```

```
router ospf 1
```

```
redistribute rip metric 50 subnets
```

```
network 172.16.1.0 0.0.0.255 area 0
```

```
!
```

```
router rip
```

```
redistribute ospf 1 metric 2
```

passive-interface Ethernet0

network 172.16.0.0

!

ipv6 router rip ripdomain

redistribute ospf 1 metric 2

ipv6 router ospf

redistribute rip ripdomain metric 50

ip classless

ip route 172.16.1.128 255.255.255.224 Ethernet0

2. RTB的配置为:

ipv6 router rip ripdomain

redistribute ospf 1 metric 2

ipv6 router ospf

redistribute rip ripdomain metric 50

summary-prefix 2001:db8:0:200::/48

3. RTB的配置为:

interface Ethernet0

ip address 172.16.1.146 255.255.255.240

ip router isis

ipv6 address 2001:db8:0:190::2/64

ipv6 router isis

!

interface Ethernet1

ip address 172.16.1.98 255.255.255.224

ip summary-address eigrp 1 172.16.1.128 255.255.255.128

ipv6 address 2001:db8:0:160::2/64

ipv6 rip ripdomain enable

!

router eigrp 1

redistribute isis level-1 metric 10000 1000 255 1 1500

passive-interface Ethernet0

network 172.16.0.0

!

router isis

net 00.0000.1234.abcd.00

summary-address 172.16.2.0 255.255.255.0

redistribute eigrp 1 metric 20 metric-type external level-1

address-family ipv6

summary-prefix 2001:db8:0:200::/48

redistribute rip metric 20 metric-type external level-1

!

ipv6 router rip ripdomain

redistribute isis metric 5

第13章

1. RTA的配置为:

```
router rip
 redistribute eigrp 1 metric 3
 passive-interface Ethernet0
 passive-interface Ethernet1
 network 172.16.0.0
 distribute-list 1 in Ethernet3
!
router eigrp 1
 redistribute rip metric 10000 1000 255 1 1500
 passive-interface Ethernet2
 passive-interface Ethernet3
 network 172.16.0.0!
 access-list 1 deny 172.16.12.0
 access-list 1 permit any
```

2. RTA的配置为:

```
router rip
 redistribute eigrp 1 metric 3
```

```
passive-interface Ethernet0
passive-interface Ethernet1
network 172.16.0.0
distribute-list 2 out Ethernet2
!
router eigrp 1
redistribute rip metric 10000 1000 255 1 1500
passive-interface Ethernet2
passive-interface Ethernet3
network 172.16.0.0
!
access-list 2 deny 172.16.10.0
access-list 2 permit any
```

3. RTA的配置为:

```
router rip
redistribute eigrp 1 metric 3
passive-interface Ethernet0
passive-interface Ethernet1
network 172.16.0.0
distribute-list 3 out eigrp 1
```

!

router eigrp 1

redistribute rip metric 10000 1000 255 1 1500

passive-interface Ethernet2

passive-interface Ethernet3

network 172.16.0.0

!

access-list 3 permit 172.16.2.0

access-list 3 permit 172.16.8.0

access-list 3 permit 172.16.9.0

4. RTA的配置为:

router rip

redistribute eigrp 1 metric 3

passive-interface Ethernet0

passive-interface Ethernet1

network 172.16.0.0

!

router eigrp 1

redistribute rip metric 10000 1000 255 1 1500

passive-interface Ethernet2

```
passive-interface Ethernet3  
network 172.16.0.0  
distribute-list 4 out Ethernet0  
!
```

```
access-list 4 permit 172.16.1.0  
access-list 4 permit 172.16.2.0  
access-list 4 permit 172.16.3.0  
access-list 4 permit 172.16.7.0  
access-list 4 permit 172.16.8.0  
access-list 4 permit 172.16.9.0
```

5. 在RIPng进程下配置的分配列表为:

```
ipv6 router rip ripname  
distribute-list prefix-list listname out ethernet2  
!  
ipv6 prefix-list listname seq 5 deny 2001:DB8:0:A::/63 le 64  
ipv6 prefix-list listname seq 10 permit ::/0 le 128
```

6. EIGRP为外部路由分配高距离值170，给内部路由分配低距离值90。如果EIGRP域内的任何目标地址从IS-IS被重新分配至EIGRP，将会被忽略，除非内部路由失效。所以，不要求手动操作EIGRP距离。RTC和RTD的IS-IS配置中的**distance** 语句为:

RTC

distance 115

distance 170 192.168.10.254 0.0.0.0 1

!

access-list 1 permit any

RTD

distance 115

distance 170 192.168.10.249 0.0.0.0 1

distance 170 192.168.10.241 0.0.0.0 1

!

access-list 1 permit any

7. RTD中IS-IS配置的**distance** 语句是：

distance 115

distance 255 192.168.10.241 0.0.0.0 1

!

access-list 1 permit any

8. RTC中EIGRP配置的**distance** 语句是：

distance eigrp 90 90

第14章

1. RTA的配置为:

```
interface Serial0  
  
ip address 172.16.14.6 255.255.255.252  
  
ip policy route-map Exercise1  
  
!  
  
interface Serial1  
  
ip address 172.16.14.10 255.255.255.252  
  
ip policy route-map Exercise1  
  
!  
  
access-list 1 permit 172.16.1.0 0.0.0.127  
  
access-list 2 permit 172.16.1.128 0.0.0.127  
  
!  
  
route-map Exercise1 permit 10  
  
match ip address 1  
  
set ip next-hop 172.16.14.17  
  
!  
  
route-map Exercise1 permit 20  
  
match ip address 2
```


set ip next-hop 172.16.14.13

2. RTA的配置为:

interface Serial0

ip address 172.16.14.6 255.255.255.252

ip policy route-map Exercise2A

!

interface Serial1

ip address 172.16.14.10 255.255.255.252

ip policy route-map Exercise2B

!

access-list 1 permit 172.16.1.64 0.0.0.63

!

route-map Exercise2 permit 10

match ip address 1

set ip next-hop 172.16.14.13

!

route-map Exercise2B permit 10

match ip address 1

set ip next-hop 172.16.14.17

3. RTA的配置为:

interface Serial2

ip address 172.16.14.18 255.255.255.252

ip access-group 101 in

ip policy route-map Exercise3

!

interface Serial3

ip address 172.16.14.14 255.255.255.252

ip access-group 101 in

ip policy route-map Exercise3

!

access-list 101 permit udp any 172.168.1.0 0.0.0.255

access-list 101 permit tcp any eq smtp 172.16.1.0 0.0.0.255

access-list 102 permit tcp any eq smtp 172.16.1.0 0.0.0.255

access-list 103 permit udp any 172.168.1.0 0.0.0.255

!

route-map Exercise3 permit 10

match ip address 102

set ip next-hop 172.16.14.9

!

route-map Exercise3 permit 20

match ip address 103

set ip next-hop 172.16.14.5

4. OSPF配置为:

router ospf 1

redistribute eigrp 1 route-map Exercise4

network 192.168.1.0 0.0.0.255 area 16

!

access-list 1 permit 10.201.100.0

!

route-map Exercise4 deny 10

match ip address 1

!

route-map Exercise4 permit 20

match route-type internal

set metric 10

set metric-type type-1

!

route-map Exercise4 permit 30

match route-type external

set metric 50

set metric-type type-2

5. EIGRP配置为:

router eigrp 1

redistribute ospf 1 route-map Exercise5

network 192.168.100.0

!

access-list 1 permit 192.168.1.0

access-list 1 permit 192.168.2.0

access-list 1 permit 192.168.3.0

!

route-map Exercise5 permit 10

match ip address 1

match route-type internal

set metric 10000 100 255 1 1500

!

route-map Exercise5 permit 30

match ip address 1

match route-type external

set metric 10000 10000 255 1 1500

附录F

故障诊断练习答案

第1章

1. 子网：10.14.64.0

主机地址：10.14.64.1～10.14.95.254

广播地址：10.14.95.255

子网：172.25.0.224

主机地址：172.25.0.225～175.25.0.254

广播地址：172.25.0.255

子网：172.25.16.0

主机地址：172.25.16.1～172.25.16.126

广播地址：172.25.16.127

2. 192.168.13.175/28是子网192.168.13.160/28的广播地址。

第3章

1. 从Piglet到子网192.168.1.64/27不再可达，从Piglet到子网10.4.6.0/24和10.4.7.0/24也不再可达。

2. 错误发生在：

- RTA，第2个表项，应该是**ip route 172.20.80.0255.255.240.0 172.20.20.1**。
- RTB，第3个表项，应该是**ip route 172.20.208.0255.255.240.0 172.20.16.50**。
- RTC，第2个表项，应该是**ip route 172.20.208.0255.255.240.0 172.20.16.50** 以及第5个表项，应该是**ip route 172.20.80.0 255.255.240.0 172.20.20.1**。

3. 错误是：

- RTC：到10.5.8.0/24的路由指向了错误的下一跳地址。
- RTC：到10.1.1.0/24的路由应该是10.5.1.0/24（网络中没有10.1.1.0/24）。
- RTC：到10.5.4.0/24没有路由。
- RTD：到10.4.5.0/24的路由应该是10.5.4.0/24。

第5章

1. 新的访问列表给每一条路由（除了10.33.32.0）添加了两跳。
2. RTB根据其错误配置的掩码解释172.16.0.0的所有子网。结果是，正如其路由表中的4个表项所示：

- 表项1：正确。因为172.16.24.0掩码既可以是22位也可以是23位。
- 表项2：RTC宣告子网172.16.26.0。因为该地址的第23位是1且RTB使用22位的掩码，从RTB的角度来看这个1出现在主机地址部分。所以，RTB把172.16.26.0的宣告看作是一条主机路由，并且在路由表里将其掩码设为32位。
- 表项3：RTB将其接口地址172.16.22.5解释为子网172.16.20.0/22的一部分，而不是172.16.22.0/23的一部分。当RTB收到RTA关于子网172.16.20.0/23的宣告时，由于RTB认为自己与该子网直接相连，于是忽略了该通告。注意，RTA和RTC的路由表里没有子网172.16.22.0，这是由于同一个原因：RTB宣告它为172.16.20.0。
- RTB将其接口地址172.16.18.4解释为子网172.16.16.0/22的一个成员，而不是172.16.18.0/23。

3. 答案在示例5-30中RTC的路由表里。注意到172.16.26.0/23的路由在2分42秒内没有更新。RTC的无效计时器在它听到来自RTD的一条新的更新之前就超时了，于是它宣布到172.16.26.0/23的路由无效。因为没有从RTA或RTB来的路由被宣布无效，所以问题出在RTD的update计时器上——更新周期太长。当RTD最终送出一条更新，它再次出现在RTC的路由表里，并且保留到RTC的无效计时器再次超时。

第6章

1. RTA和RTB只发送和接收RIPv2消息。RTC接受RIPv1和RIPv2消息，但只发送RIPv1消息。结果是，RTA和RTB的路由表里没有子网192.168.13.75/27。虽然RTC接收来自RTA和RTB的更新，但它的路由表里没有子网192.168.13.90/28和192.168.13.86/29，因为RTC把它们解释成192.168.13.64/27，这是与RTC的E0接口直接相连的。
2. 是的。RTA和RTB的路由表中将会加入子网192.168.13.64/27，因为它们现在能接收来自RTC的RIPv1更新。

第7章

1. 在到子网A的路径上RTG是RTF的后继。
2. RTC到子网A的可行距离是309 760。
3. RTG到子网A的可行距离是2 214 319。
4. RTG的拓扑表显示，RTD和RTE是它到子网A的可行后继。
5. RTA到子网B的可行距离是2 198 016。

第8章

1. 在一个邻居接口上，IP地址或者其掩码配置出错。
2. 一台路由器被配置为末梢区域路由器，另一台不是。
3. 接收端路由器被配置为MD5（类型2）验证，其邻居没有配置认证（类型0）。
4. 两台路由器上配置的密码不一致。
5. 邻居接口没有被配置相同的区域ID。
6. RTA的**network** 语句顺序错误。第一个语句匹配IP地址192.168.50.242，并把它放入区域192.168.50.0。
7. 链路ID 10.8.5.1最可疑，因为它的序列号比其他链路的序列号要高很多。

第10章

1. 虽然IS-IS形成了邻接关系，但是它的类型是IS而不是L1、L2或者L1/L2，这表明存在问题。IP地址并不在同一个子网里。
2. 路由器只发送L1 Hello，表明它是一台L1路由器。它只从0000.3090.c7df接收L2 Hello，表明它是一台L2路由器。

第11章

1. 水平分割将阻止192.168.10.0/24从IGRP域到RIP域的宣告。
2. 不。因为不是所有10.0.0.0的子网都在RIP端与Mantle直接相连。
3. Robinson的EIGRP1配置中的**network** 语句匹配接口192.168.3.32，即使没有EIGRP的Hello在这个接口被转发出去。因此，这个子网在EIGRP1域内被声明为内部的。
4. EIGRP自动加入这条汇总路由，因为子网192.168.3.0是从OSPF重新分配到EIGRP中的。
5. 区域1中的第1层路由器将不知道汇总路由。RIP被重新分配到第2层中（这是缺省 情况进行分配的层）。命令**summary-prefix** 正在汇总被通告到第1层中的路由。

第13章

1. 访问列表第一行中的反码错误。由于此反码，所有路由将被匹配。这一行应该是**access-list 1 deny 0.0.0.0 0.0.0.0**。
2. 所有没有被访问列表匹配的路由将赋予距离值255（不可到达）。如果一个优先路由失败，Grimwig将不使用备用路径。
3. OSPF通过它从LSA学来的信息计算路由。路由过滤器不影响LSA，所以过滤器仅仅影响配置它的路由器。
4. 这两台路由过滤器涉及错误的接口。**distribute-list 1** 应该对应E1, **distribute-list 2** 应该对应E0。

第14章